Confluence and Preservation of Strong Normalisation in an Explicit Substitutions Calculus

César Augusto Munoz Hurtado

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET AUTOMATIQUE

Confluence and preservation of strong normalisation in an explicit substitutions calculus

César Augusto Muñoz Hurtado

N ° 2762

Décembre 1995

PROGRAMME 2

Calcul symbolique,
programmation
et génie logiciel

ISSN 0249-6399

Confluence and preservation of strong normalisation in an explicit substitutions calculus

César Augusto Muñoz Hurtado*

Programme 2 — Calcul symbolique, programmation et génie logiciel

Projet Coq

Rapport de recherche n°2762 — Décembre 1995 — 42 pages

Abstract: Explicit substitutions calculi are formal systems that implement β-reduction by means of an internal substitution operator. Thus, in that calculi it is possible to delay the application of a substitution to a term or to consider terms with partially applied substitutions. This feature is useful, for instance, to represent incomplete proofs in type based proof systems. The λ_σ-calculus of explicit substitutions proposed by Abadi, Cardelli, Curien and Lévy gives an elegant way to deal with management of variable names and substitutions of λ-calculus. However, λ_σ does not preserve strong normalisation of λ-calculus and it is not a confluent system. Typed variants of λ_σ without composition are strongly normalising but not confluent, while variants with composition are confluent but do not preserve strong normalisation. Neither of them enjoys both properties. In this paper we propose the λ_ζ-calculus an we present the full proofs of its main properties. This is, as far as we know, the first confluent calculus of explicit substitutions that preserves strong normalisation.

Key-words: Explicit substitutions, confluence, preservation of strong normalisation, λ-calculus

(Résumé : tsvp)

*Cesar.Munoz@inria.fr

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
Téléphone : (33 1) 39 63 55 11 – Télécopie : (33 1) 39 63 53 30

Confluence et préservation de la normalisation forte dans un calcul avec substitutions explicites

Résumé : Les calculs avec substitutions explicites sont des systèmes formels où la β-réduction est définie en utilisant des opérateurs de substitution interne à la théorie. Donc, dans un tel calcul, on peut retarder l'application d'une substitution à un terme et considérer des termes avec substitutions partiellement appliqués. De telles caractéristiques sont utiles pour la représentation de preuves incomplètes dans un système de traitement de preuves basé sur la théorie des types. Le calcul avec substitutions explicites λ_σ, proposé par Abadi, Cardelli, Curien et Lévy, fournit une façon élégante de manipuler les noms de variables et les substitutions du λ-calcul. Mais λ_σ ne préserve pas la normalisation forte et n'est pas un système confluent. Plusieurs présentations alternatives qui satisfont la première ou la seconde propriété on été proposés, mais aucune d'entre elles ne satisfait les deux propriétés. Dans cet article on propose un calcul appelé λ_ζ-calcul et on présente les preuves complètes de ses principaux propriétés. Ce calcul est à notre connaissance le premier calcul confluent avec substitutions explicites et qui préserve la normalisation forte.

Mots-clé : Substitutions explicites, confluence, préservation de la normalisation forte, λ-calcul

1 Introduction

In the traditional λ-calculus, the β-reduction, $(\lambda x.M)N \xrightarrow{\beta} M\{x \leftarrow N\}$, is expressed by using a substitution operation treated as an atomic step and defined outside the theory. Thus in λ-calculus it is not possible to delay the application of a substitution to a term or to consider terms with partially applied substitutions. This feature is useful, for instance, to represent incomplete proofs in type based proof systems.

Explicit substitutions calculi are formal systems that implement β-reductions by means of an internal substitution operator. The λ_σ-calculus of explicit substitutions, introduced by [ACCL91], gives an elegant way to deal with management of variable names and substitutions of λ-calculus. It consists of a set of terms that uses the de Bruijn's notation of variables ([dB72]) in order to deal with α-conversion (renaming of bound variables), and a first order rewriting system that implements the substitution operation.

However, λ_σ lacks some properties of λ-calculus and, moreover, it raises new theoretical problems. First, the typed version of λ_σ is not terminating ([Mel95]) and second, λ_σ introduces the problem of confluence on open terms.

In calculus with the de Bruijn's notation (for example λ_σ), both closed terms (e.g. $\lambda x.x$) and open terms (e.g. $\lambda x.y$) are coded as de Bruijn's ground terms ($\lambda\underline{1}$ and $\lambda\underline{2}$, respectively). Thus the usual confluence property of λ-calculus corresponds to confluence on de Bruijn's ground terms. The variables of the first order theory[1], which stand for arbitrary λ-terms, are not handled directly by classical λ-calculus. For example, if X and Y denote arbitrary λ-terms, the term $(\lambda X)Y$ is not reducible until we know what X and Y are (even, if this term is a β-redex). In λ_σ, this reduction is possible because the term X annotated with the substitution $\{\underline{1} \leftarrow Y\}$ is a valid λ_σ-term. Unfortunately, λ_σ is not a confluent first-order system ([CHL95]), i.e. it is not confluent on terms which may contain meta-variables. It is only confluent on ground terms ([ACCL91]) and on substitution-ground terms ([Río93]), i.e. on terms without substitution meta-variables.

[CHL95] explores another calculus of explicit substitutions, namely $\lambda_{\Uparrow}$, with the confluence property[2]. The innovation of that presentation is the substitution operator "lift" ($\Uparrow$[3]), which is used to introduce a substitution inside an abstraction in order to avoid the harmful critical pair which leads to the non-confluence in λ_σ. However, the non-termination problem of its simply typed version still subsists.

On the other hand, [Les94] proposes the λ_υ system. This calculus has a simple presentation (a small number of rules) and, nicely, it preserves the strong normalisation, i.e. every term strongly β-normalisable (particularly, every typed λ-term) is strongly λ_υ-normalisable. In order to avoid non terminating chains of reductions, this system lack for rules of substitution composition. Thus, λ_υ is not a confluent system, even if it is confluent on ground terms.

Apparently, there is a choice: confluence (and only weakly normalisation in typed terms) or strong normalisation in typed terms (and only ground confluence).

[1] Usually they are called *meta-variables* to distinguish from the variables written as de Bruijn's indices.
[2] In the following confluence means "confluence in the first-order" sense, i.e. on terms with meta-variables.
[3] If S is a substitution, $\Uparrow(S)$ denotes the substitution $\mathbf{1} \cdot (S \circ \uparrow)$

The confluence is interesting (and perhaps necessary) to achieve the higher-order unification via explicit substitutions (see [DHK95]) or to deal with existential variables in type based proof system (see [Mag94]). On the other hand, the strong normalisation property is interesting to implement traditional typed systems.

We propose a confluent calculus of explicit substitutions which preserves strong normalisation, system λ_ζ, and we prove its main properties. The paper is structured as follows. Section 2 explains some problems related to termination and confluence in explicit substitutions calculi. Section 3 introduces the system λ_ζ. Section 4 relates λ_ζ to the classical λ-calculus. Section 5 and 6 address, respectively, the proofs of confluence and preservation of strong normalisation of the system λ_ζ, and Section 7 provides the conclusions of this work and discuss related open problems.

2 The Problems

We recall some usual concepts related to explicit substitutions calculi.

In the de Bruijn's notation, the binding occurrences of variables in λ-constructions are dropped, and each occurrence of a variable in a term is replaced by a natural number (its de Bruijn's index). The index $\underline{n}$ is captured by the n-th λ-constructor surrounding it. For example the term $\lambda x.((\lambda y.x\ y)x)$ becomes $\lambda((\lambda\underline{2}\ \underline{1})\underline{1})$.

Free variables are represented by natural numbers also, but they are referenced with respect to a context of free variables. If there are m λ-constructors surrounding the index $\underline{n}$, and $n > m$, then $\underline{n}$ represent the $(n-m)$-th free variable of the context. For example, the term $\lambda x.(y\ z)$ becomes $\lambda(\underline{2}\ \underline{3})$ in the context (y, z).

A substitution can be seen as an infinite sequence of terms $M_1, M_2, \ldots, M_n, \ldots$, with the intuitive meaning $\{\underline{1} \leftarrow M_1, \underline{2} \leftarrow M_2, \ldots, \underline{n} \leftarrow M_n, \ldots\}$. There are two special substitutions: id which represents the identity substitution ($\{\underline{n} \leftarrow \underline{n}\}$, for every positive natural n), and the shift substitution ($\uparrow$) which represents the substitution $\{\underline{n} \leftarrow \underline{n+1}\}$, for every positive natural n.

The application of the substitutions S to a term M is denoted by $M[S]$. In practice, the only substitutions that are considered are those that can be represented by means of finite lists of terms, built with the operator *cons* $(\cdot)$ and the constants id and $\uparrow$. If M is a term and S is a substitution, then $M \cdot S$ represents the substitution $\{\underline{1} \leftarrow M, \ldots, \underline{n+1} \leftarrow \underline{n}[S] \ldots\}$, for every positive natural n.

Moreover, if S, T are substitutions and M is a term:

- $S \circ T$ denotes the *composition* of S and T, defined as $\{\underline{n} \leftarrow M[T]\}$, for any $\underline{n} \leftarrow M$ in S,
- $\Uparrow (S)$ denotes the *lift* of S, defined as $\{\underline{1} \leftarrow \underline{1}, \underline{n+1} \leftarrow \underline{n}[S][\uparrow]\}$, for every positive natural n, and
- $M/$ [4] denotes the substitution $M \cdot id$.

[4] The operator / was introduced originally in [Ehr88] and used by [Río93] in the system λ_τ.

$(\lambda M)M'$	$\longrightarrow$	$M[M' \cdot id]$	(Beta)
$(M\ M')[S]$	$\longrightarrow$	$M[S]M'[S]$	(Application)
$(\lambda M)[S]$	$\longrightarrow$	$\lambda(M[\Uparrow(S)])$	(Lambda)
$M[S][S']$	$\longrightarrow$	$M[S \circ S']$	(Clos)
$\mathbf{1}[M \cdot S]$	$\longrightarrow$	M	(VarCons)
$\mathbf{1}[\Uparrow(S)]$	$\longrightarrow$	$\mathbf{1}$	(VarLift1)
$\mathbf{1}[\Uparrow(S) \circ S']$	$\longrightarrow$	$\mathbf{1}[S']$	(VarLift2)
$M[id]$	$\longrightarrow$	M	(Id)
$(S \circ S') \circ S''$	$\longrightarrow$	$S \circ (S' \circ S'')$	(Ass)
$(M \cdot S) \circ S'$	$\longrightarrow$	$M[S'] \cdot (S \circ T)$	(Map)
$\Uparrow(S) \circ \Uparrow(S')$	$\longrightarrow$	$\Uparrow(S \circ S')$	(Lift1)
$\Uparrow(S) \circ (\Uparrow(S') \circ S'')$	$\longrightarrow$	$\Uparrow(S \circ S') \circ S''$	(Lift2)
$\Uparrow(S) \circ (M \cdot S')$	$\longrightarrow$	$M \cdot (S \circ S')$	(Liftcons)
$id \circ S$	$\longrightarrow$	S	(Idl)
$S \circ id$	$\longrightarrow$	S	(Idr)
$\Uparrow(id)$	$\longrightarrow$	id	(LiftId)
$\uparrow \circ M \cdot S$	$\longrightarrow$	S	(ShiftCons)
$\uparrow \circ \Uparrow(S)$	$\longrightarrow$	$S \circ \uparrow$	(ShiftLift1)
$\uparrow \circ (\Uparrow(S) \circ S')$	$\longrightarrow$	$S \circ (\uparrow \circ S')$	(ShiftLift2)

Figure 1: The rewriting system $\lambda_{\Uparrow}$

Now, we present two representative systems for each property: $\lambda_{\Uparrow}$ for confluence and λ_v for preservation of strong normalisation.

$\lambda_{\Uparrow}$ ([CHL95]) is defined by the following grammar and its rewriting system is presented in Fig. 1.

Terms	M	$::=$	$\mathbf{1} \mid \lambda M \mid M\ M' \mid M[S]$
Substitutions	S	$::=$	$id \mid \Uparrow(S) \mid \uparrow \mid M \cdot S \mid S \circ S'$

Briefly, the system uses the operator lift and the composition of substitutions. The former one allows the introduction of substitutions into the constructor λ (see rule (Lambda)) and the latter one allows to calculate with substitutions (see the rules (Ass) until (ShiftLift2)).

Also the index $\underline{1}$ is represented by $\mathbf{1}$ and $\underline{n+1}$ is represented by $\mathbf{1}[\overbrace{(\uparrow \circ \ldots \circ (\uparrow \circ \uparrow)\ldots)}^{n\text{-times}}]$, for every positive natural n.

$(\lambda M)M'$	$\longrightarrow$	$M[M'/]$	(Beta)
$(M\ M')[S]$	$\longrightarrow$	$M[S]\ M'[S]$	(Application)
$(\lambda M)[S]$	$\longrightarrow$	$\lambda(M[\Uparrow(S)])$	(Lambda)
$\underline{1}[M/]$	$\longrightarrow$	M	(FVar)
$\underline{n+1}[M/]$	$\longrightarrow$	$\underline{n}$	(RVar)
$\underline{1}[\Uparrow(S)]$	$\longrightarrow$	$\underline{1}$	(FVarLift)
$\underline{n+1}[\Uparrow(S)]$	$\longrightarrow$	$\underline{n}[S][\uparrow]$	(RVarLift)
$\underline{n}[\uparrow]$	$\longrightarrow$	$\underline{n+1}$	(VarShift)

Figure 2: The rewriting system λ_v

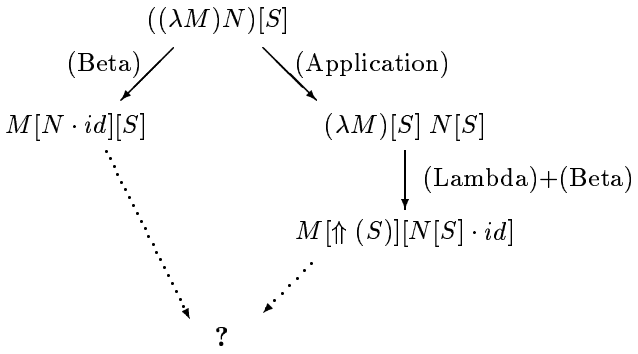
The grammar of λ_v ([Les94]) is:

Nat	n	::=	$1 \mid n+1$
Terms	M	::=	$\underline{n} \mid \lambda M \mid M\ M' \mid M[S]$
Substitutions	S	::=	$M/ \mid \uparrow \mid \Uparrow(S)$

and its rewriting system is presented in Fig. 2.

The principal difference with the system $\lambda_{\Uparrow}$, is that the system λ_v lacks rules of substitution composition, for example the term $M[\uparrow][S]$ is not a λ_v-redex. Thus, there are not simultaneous substitutions and therefore the dotted notation of lists can be changed by the more simple /-notation.

In order to carry-out the local confluence in any system which contains (Beta), (Application) and (Lambda), (for example λ_σ, $\lambda_{\Uparrow}$, λ_v and λ_τ [Río93]) it is necessary to solve the critical pair



In [LRD94], the Substitutions Lemma proves that the critical pair is solved in the λ_v-rewriting system when M is a ground term. But to achieve local confluence on open terms, λ_σ and $\lambda_{\Uparrow}$ introduce an operator to compose substitutions allowing to close this critical pair on the term $M[N[S] \cdot S]$. This operator being apparently responsible for non termination, we have taken another approach: we cut one of the branches of the above critical pair.

To implement this idea, we have two natural strategies:

1. cut the right branch, i.e. the reduction of the β-redex before the distribution of the substitution, and
2. cut the left branch, i.e. the distribution of the substitution before the reduction of the β-redex.

To cut the right branch we must avoid the distribution of a substitutions inside a β-redex. Thus it is necessary to change the (Application) rule by a rule like ([Her95])

$$(\dots((\underline{n}\ M_1)M_2)\dots M_m)[S] \qquad \longrightarrow \qquad (\dots((\underline{n}[S]\ M_1[S])M_2[S])\dots M_m[S]) \quad \text{(Application')}$$

Indeed, with this rule the distribution of substitutions to applications is only possible if the head term is a variable. Thus, in the above critical pair, the substitution S is not propagated into $((\lambda M)N)$ because the header of the term is not a variable. Therefore, only one reduction rule applies and there is no critical pair.

But this rule is not a first order rule, because the header of the term can be at an arbitrary depth, thus if we want to code it directly in a explicit substitutions calculus we must change the structure of application from $(\dots(M M_1)\dots M_n)$ to $M\{M_1\dots M_n\}$, in order to deal with a list of arguments, just like in the $\overline{\lambda_\sigma}$-version ([Her94]) of the (Application) rule:

$$(\underline{n}\ L)[S] \qquad \longrightarrow \qquad \underline{n}[S]\ L[S] \quad \text{(Application-List)}$$

Here L is a list of terms which are the arguments of the head variable $\underline{n}$.

The implementation of the second strategy, i.e. to cut the left branch of the critical pair, by a first order rewriting system remains as an open problem.

The main idea of the system λ_ζ is to cut the right branch of the above critical pair by coding the (Application-list) rule in the usual notation for application.

Thus, we want the following reduction: $((\underline{n}\ M_1)M_2)[S] \xrightarrow{\lambda_\zeta^*} (\underline{n}[S]\ M_1[S])\ M_2[S]$ but we want to forbid the following one: $(((\lambda M)M_1)M_2)[S] \xrightarrow{*} ((\lambda M)[S]\ M_1[S])\ M_2[S]$.

In order to get a confluent first order rewriting system we shall introduce two types of marked applications (or simply *marks*): $\bullet$-marks and $\odot$-marks. Unmarked applications become $\bullet$-marks if they are applied to substitutions. $\bullet$-marks become $\odot$-marks if they have a head variable. Finally, the distribution of substitutions is only possible in $\odot$-marks.

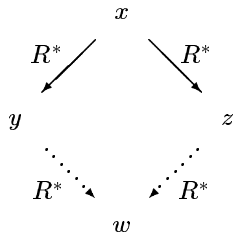
In the presentation that follows, we need some abstract properties about relations defined on a set X. We recall a few general concepts and lemmas.

Notation

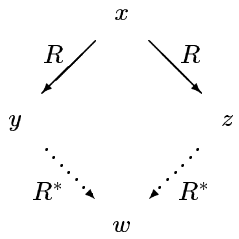
Let R and S be two relations defined on the set X, and x, y two elements of X:

- RS is the composition of R with S.
- R^* is the transitive-reflexive closure of R.
- R^n is the n-times composition of R.
- $x \xrightarrow{R} y$ is the same that: $(x, y) \in R$. Usually we read this notation as "x reduces to y by R".
- $x \overset{R}{\cdots\!\!\rightarrow} y$ is the same that: There exists a y such that $x \xrightarrow{R} y$

Definition 1 (Confluence) *A relation R is* confluent *if and only if for any x, y, z in X such that $x \xrightarrow{R^*} y$ and $x \xrightarrow{R^*} z$, there exists w in X such that $y \xrightarrow{R^*} w$ and $z \xrightarrow{R^*} w$, i.e. the following diagram holds:*

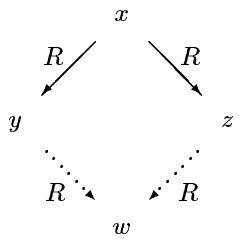


Definition 2 (Locally Confluence) *A relation R is* locally confluent *if and only if for any x, y, z in X such that $x \xrightarrow{R} y$ and $x \xrightarrow{R} z$, there exists w in X such that $y \xrightarrow{R^*} w$ and $z \xrightarrow{R^*} w$, i.e. the following diagram holds:*



Definition 3 (Strong Confluence) *A relation R is* strongly confluent *if and only if for any x, y, z in X such that $x \xrightarrow{R} y$ and $x \xrightarrow{R} z$, there exists w in X such that $y \xrightarrow{R} w$*

and $z \xrightarrow{R} w$*, i.e. the following diagram holds:*



Definition 4 (Normal Form) *An element* x *of* X *is a* R-normal form *if and only if there is no* y *in* X *such that* $x \xrightarrow{R} y$.

Definition 5 (Normalisation) *An element* x *of* X *is* R-normalisable *if and only if there exists a* R*-normal form* y *such that* $x \xrightarrow{R^*} y$.

Definition 6 (Strong Normalisation) *An element* x *of* X *is* strongly R-normalisable *if and only if any* R*-reduction of* x *is finite.*

Definition 7 (Termination) *A relation* R *is terminating if and only if* x *is strongly* R*-normalisable for every* x *in* X.

Lemma 1 (Newman's lemma) *A terminating relation is confluent if and only if it is locally confluent.*

Proof. See [Hue80]. □

Lemma 2 *Any strongly confluent relation is confluent.*

Proof. See [Hue80]. □

Lemma 3 *If* R *is confluent, then the* R*-normal form of any element, if it exists, is unique.*

Proof. See [Hue80]. □

3 The System λ_ζ

The system λ_ζ is similar to λ_υ in the sense that substitutions are not composed and it preserves strong normalisation too; but in contrast to λ_υ, λ_ζ is fully confluent (not only in ground terms). It is defined by the following grammar and the rewriting system of Fig. 3:

Nat	n	::=	$1 \mid n+1$
terms	M	::=	$\underline{n} \mid \lambda M \mid M\ M' \mid M[S] \mid M \bullet M' \mid M \odot M'$
Substitutions	S	::=	$M/ \mid \uparrow \mid \Uparrow (S)$

$(\lambda M)N$	$\longrightarrow$	$M[N/]$	(Beta)
$(\lambda M) \bullet N$	$\longrightarrow$	$M[N/]$	(Beta$^\bullet$)
$(\lambda M)[S]$	$\longrightarrow$	$\lambda(M[\Uparrow (S)])$	(Lambda)
$\underline{1}[M/]$	$\longrightarrow$	M	(FVar)
$\underline{n+1}[M/]$	$\longrightarrow$	$\underline{n}$	(RVar)
$\underline{1}[\Uparrow (S)]$	$\longrightarrow$	$\underline{1}$	(FVarLift)
$\underline{n+1}[\Uparrow (S)]$	$\longrightarrow$	$\underline{n}[S][\uparrow]$	(RVarLift)
$\underline{n}[\uparrow]$	$\longrightarrow$	$\underline{n+1}$	(VarShift)
$(M \odot N)[S]$	$\longrightarrow$	$M[S]\ N[S]$	(UnMark)
$(MN)[S]$	$\longrightarrow$	$(M \bullet N)[S]$	(Mark)
$\underline{n} \bullet M$	$\longrightarrow$	$\underline{n} \odot M$	(ReMark)
$(M\ N) \bullet M'$	$\longrightarrow$	$(M \bullet N) \bullet M'$	(Application$^\bullet$)
$(M \odot N) \bullet M'$	$\longrightarrow$	$(M \odot N) \odot M'$	(Application$^\odot$)

Figure 3: The rewriting system λ_ζ

Notice that the original rule (Application) becomes (Mark) in our system and we have added two marked application's operators: $\bullet$ and $\odot$. These operators are necessary to introduce substitutions inside applications only if the application has a head variable. We can see the use of marks as a hand-shaking process between substitutions and head variables. The $\bullet$-mark is a request for a head variable in the term (rule (Mark)) and it is propagated recursively into the left part of the application (rule (Application$^\bullet$)). The head variable answers positively to a $\bullet$-mark with a $\odot$-mark (rule (ReMark)), which is propagated out of term erasing the $\bullet$-marks (rule (Application$^\odot$)). The hand-shaking is completed when substitutions go through the $\odot$-marks erasing them (rule (UnMark)). If a $\bullet$-mark is not propagated into an application, for example because there is a λ, then it is erased by the (Beta$^\bullet$) rule. In λ_ζ, substitution redex disappears using the same rules as in λ_υ. Thus we have the following reductions:

$$((\underline{n}\ M_1)M_2)[S] \xrightarrow{\lambda_\zeta^*} ((\underline{n} \bullet M_1) \bullet M_2)[S] \xrightarrow{\lambda_\zeta^*} ((\underline{n} \odot M_1) \odot M_2)[S] \xrightarrow{\lambda_\zeta^*} (\underline{n}[S]\ M_1[S])\ M_2[S]$$

But, $(((\lambda M)M_1)M_2)[S] \xrightarrow{\lambda_\zeta^*} (((\lambda M) \bullet M_1) \bullet M_2)[S]$, thus to apply the substitution S, we must first reduce $((\lambda M) \bullet M_1) \bullet M_2$ in order to have a term with a head variable and so a $\odot$-mark.

We define some useful subsystems of λ_ζ:

Definition 8 *Consider the rewriting system λ_ζ, then:*

- *the system $B^\bullet$ is the system formed by the rules* (Beta) *and* (Beta$^\bullet$),
- *the system ζ is the system obtained by dropping the rules* (Beta) *and* (Beta$^\bullet$), *and*
- *the system $\mathcal{M}$ is the system formed by the rules* (Mark), (Remark), (Application$^\bullet$) *and* (Application$^\odot$).

Lemma 4 *For any ground terms M, M' and ground substitution S, neither $M \bullet M'$, nor $M[S]$, are λ_ζ-normal forms.*

Proof. If M, M' or S are not normal forms, then the result is obvious. Otherwise, we prove by simultaneous structural induction that: (1) $M \bullet M'$ is not a normal form, and (2) $M[S]$ is not a normal form.

- If $M = n$ then (1) $n \bullet M'$ is reducible by the rule (ReMark), and (2) $M[S]$ is reducible by (FVar), (RVar), (FVarLift), (RVarLift) or (VarShift).
- If $M = \lambda N$ then we reduce (1) with (Beta$^\bullet$) and (2) with (Lambda).
- If $M = NN'$ then we reduce (1) with (Application$^\bullet$) and (2) with (Mark).
- If $M = N \odot N'$ then we reduce (1) with (Application$^\odot$) and (2) with (UnMark).
- By induction hypothesis M is either $N \bullet N'$ nor $N[T]$.

□

Lemma 5 *The set of ground λ_ζ-normal forms is described by:*

Normal form	$\widehat{M}$	$::=$	$\widehat{H} \mid \lambda\widehat{M}$
Applicative normal form	$\widehat{H}$	$::=$	$\underline{n} \mid \widehat{H}\widehat{M} \mid \widehat{M} \odot \widehat{M'}$
Substitution normal form	$\widehat{S}$	$::=$	$\widehat{M}/ \mid \uparrow \mid \Uparrow(\widehat{S})$

Proof. First, we check that all the terms and substitutions defined by this grammar are in normal form. Then, we use Lemma 4 to prove by structural induction that all the normal form have the specified form. □

4 Relation with classical λ-calculus

We say that a λ_ζ-term is *pure* if it does not contain neither marks nor substitutions. Therefore the set of pure λ_ζ-terms is the set of ground λ-terms represented in the de Bruijn's notation. Notice that the set of normal λ_ζ-terms is not included in the set of ground λ-terms. For example, the normal form $\underline{1} \odot \underline{1}$ is not pure. However we can prove that the marks disappear in normal λ_ζ-reductions.

The propositions that reflect the interpretation of β-reduction in the λ_ζ-system are:

Proposition 1 (Soundness) *If M is a pure term and $M \xrightarrow{\lambda_\zeta^*} \widehat{M}$ is a λ_ζ-normal reduction, then $M \xrightarrow{\beta^*} \widehat{M}$ is a β-normal reduction too.*

Proposition 2 (Completeness) *If M is a pure term and $M \xrightarrow{\beta^*} \widehat{M}$ is a β-normal reduction, then $M \xrightarrow{\lambda_\zeta^*} \widehat{M}$ is a λ_ζ-normal reduction too.*

This semantic of β-reduction is called "big step", in contrast with "one step" semantic which corresponds to the simulation of only one step of β-reduction. It is easy to see that "one step" semantic is not strictly implemented in λ_ζ. For example the reduction $\overbrace{(\lambda((\lambda\underline{1})\underline{2}))\underline{1}}^{\beta\text{-redex}} \xrightarrow{\beta} (\lambda\underline{1})\underline{1}$ is not possible in λ_ζ, because $\overbrace{(\lambda((\lambda\underline{1})\underline{2}))\underline{1}}^{(\text{Beta})\text{-redex}}$ reduces to $((\lambda\underline{1})\underline{2})[\underline{1}/]$ in the λ_ζ-system, and we cannot perform the substitution until the redex $(\lambda\underline{1})\underline{2}$ is not reduced. Thus one step reduction is not preserved unless we consider $((\lambda\underline{1})\underline{2})[\underline{1}/]$ as equivalent to $(\lambda\underline{1})\underline{1}$ modulo substitutions. In contrast, in both systems the following normal reduction is possible: $(\lambda(\overbrace{(\lambda\underline{1})\underline{2}}^{\beta\text{-redex}}))\underline{1} \xrightarrow{*} \overbrace{(\lambda\underline{2})\underline{1}}^{\beta\text{-redex}} \xrightarrow{*} \underline{1}$.

Now we address the proofs of the above propositions.

4.1 λ_ζ-Soundness

Definition 9 *We define the following subsets of λ_ζ-terms:*

- *An λ_ζ-term is* unmarked *if and only if it is a ground term and it does not contain marks.*
- *An λ_ζ-term is* pure *if and only if it is unmarked and it does not contain substitutions.*
- *The set of purifiable terms is defined by the grammar:*

Purifiable terms	P	$::=$	$\underline{n} \mid \lambda P \mid P\,P' \mid \overset{\bullet}{P}[S]$
Purifiable$^\bullet$ terms	$\overset{\bullet}{P}$	$::=$	$P \mid \overset{\odot}{P} \mid \overset{\bullet}{P} \bullet P$
Purifiable$^\odot$ terms	$\overset{\odot}{P}$	$::=$	$\underline{n} \mid \overset{\odot}{P} \odot P$
Purifiable substitutions	S	$::=$	$P/ \mid \uparrow \mid \Uparrow(S)$

Notice that: Pure Terms $\subseteq$ Unmarked Terms $\subseteq$ Purifiable Terms $\subseteq \lambda_\zeta$-terms, but Normal Terms $\not\subseteq$ Pure Terms. We will prove that marks disappear in normal forms of purifiable terms (and therefore, in normal forms of pure and unmarked terms).

Lemma 6 *Let x and y be two λ_ζ-terms such that $x \xrightarrow{\lambda_\zeta} y$, then*

- *if x is a purifiable term then y is a purifiable term,*

- *if x is a purifiable$^\bullet$ term then y is a purifiable$^\bullet$ term,*
- *if x is a purifiable$^\odot$ term then y is a purifiable$^\odot$ term, and*
- *if x is a purifiable substitution then y is a purifiable substitution.*

Proof. By simultaneous structural induction and cases analysis in the λ_ζ-rule applied to reduce x to y. We consider only the cases that introduce or delete marks, the other are not very interesting.

- $x = (P \odot P')[S]$ and $y = P[S]\ P'[S]$ with $x \xrightarrow{\text{(UnMark)}} y$. By hypothesis x is a purifiable term and so P, P' and S are purifiable terms, then by definition $P[S]$, $P'[S]$ and y are purifiable terms too.
- $x = (P\ P')[S]$ and $y = (P \bullet P')[S]$ with $x \xrightarrow{\text{(Mark)}} y$. By hypothesis x is a purifiable term and so P, P' and S are purifiable terms, then by definition $P \bullet P'$ is a purifiable$^\bullet$ term and so y is a purifiable term.
- $x = (\lambda P) \bullet P'$ and $y = P[P'/]$ with $x \xrightarrow{\text{(Beta}^\bullet\text{)}} y$. By hypothesis x is a purifiable$^\bullet$ term and so P and P' are purifiable terms, then by definition y is a purifiable term too, and therefore by definition y is a purifiable$^\bullet$ term.
- $x = \underline{n} \bullet P$ and $y = \underline{n} \odot P$ with $x \xrightarrow{\text{(ReMark)}} y$. By hypothesis x is a purifiable$^\bullet$ term and so P is a purifiable term, then by definition y is a purifiable$^\odot$ term and therefore by definition y is a purifiable$^\bullet$ term.
- $x = (P\ P') \bullet P"$ and $y = (P \bullet P') \bullet P"$ with $x \xrightarrow{\text{(Application}^\bullet\text{)}} y$. By hypothesis x is a purifiable$^\bullet$ term and so P, P' and $P"$ are purifiable terms, then by definition y is a purifiable$^\bullet$ term.
- $x = (P \odot P') \bullet P"$ and $y = (P \odot P') \odot P"$ with $x \xrightarrow{\text{(Application}^\odot\text{)}} y$. By hypothesis x is a purifiable$^\bullet$ term and so P, P' and $P"$ are purifiable terms, then by definition y is a purifiable$^\odot$ term and therefore by definition y is a purifiable$^\bullet$ term.

□

Lemma 7 *Normal forms of purifiable terms are pure.*

Proof. Let $\widehat{P}$ be a normal form of a purifiable term P. By Lemma 6, $\widehat{P}$ is also purifiable. Since $\widehat{P}$ is a normal form, by Lemma 5, it does not contain substitutions. We verify easily that any purifiable term without substitutions is pure, and therefore we conclude that $\widehat{P}$ is a pure term. □

Corollary 1 *Normal forms of pure and unmarked terms are pure.*

A definition of β-reduction using the de Bruijn's notation is (see [CHL95] or [BBLRD95])

$$(\lambda M)N \xrightarrow{\beta} \sigma_0(M, N)$$

where σ_n is defined for every non negative number n as

$$\begin{array}{lcl} \sigma_n(M\ M', N) & = & \sigma_n(M, N)\ \sigma_n(M', N) \\ \sigma_n(\lambda M, N) & = & \lambda\sigma_{n+1}(M, N) \\ \sigma_n(\underline{m}, M) & = & \begin{cases} \underline{m-1} & \text{if } m > n+1 \\ \tau_0^n(M) & \text{if } m = n+1 \\ \underline{m} & \text{if } m \leq n \end{cases} \end{array}$$

and

$$\begin{array}{lcl} \tau_i^n(M\ N) & = & \tau_i^n(M)\ \tau_i^n(N) \\ \tau_i^n(\lambda M) & = & \lambda\tau_{i+1}^n(M) \\ \tau_i^n(\underline{m}) & = & \begin{cases} \underline{m+n} & \text{if } m > i \\ \underline{m} & \text{if } m \leq i \end{cases} \end{array}$$

Using the functions σ_n and τ_n, [BBLRD95] defines a translation μ to map ground λ_v-terms into λ-terms. We extend this function to take care of marks.

$$\begin{array}{lcl} \mu(M \bullet N) & = & \mu(M)\ \mu(N) \\ \mu(M \odot N) & = & \mu(M)\ \mu(N) \\ \mu(M\ N) & = & \mu(M)\ \mu(N) \\ \mu(\underline{n}) & = & \underline{n} \\ \mu(\lambda M) & = & \lambda\mu(M) \\ \mu(M[\Uparrow^n (N/)]) & = & \sigma_n(\mu(M), \mu(N)) \\ \mu(M[\Uparrow^n (\uparrow)]) & = & \tau_n^1(\mu(M)) \end{array}$$

Notice that if M is a pure term then $\mu(M) = M$.

Lemma 8 *Let M and N be ground λ_ζ-terms. If $M \xrightarrow{\zeta} N$ then $\mu(M) = \mu(N)$.*

Proof. The same proof for v[5] (see Proposition 1 in [BBLRD95]) extend naturally to ζ. In fact we must only consider the new rules (UnMark), (Mark), (ReMark), (Application$^\bullet$) and (Application$^\odot$). If $M \xrightarrow{\zeta} N$ using one of the (ReMark),(Application$^\bullet$) or (Application$^\odot$) rules the result is trivial from the definition of μ.

- Case (UnMark). In order to prove that $\mu((M \odot M')[S]) = \mu(M[S]\ M'[S])$, we consider two cases.
  - $S = \Uparrow^n (N/)$. We have that $\mu((M \odot M')[S]) = \sigma_n(\mu(M \odot M'), \mu(N)) = \sigma_n(\mu(M)\ \mu(M'), \mu(N)) = \sigma_n(\mu(M), \mu(N))\ \sigma_n(\mu(M'), \mu(N)) = \mu(M[\Uparrow^n (N/)])\ \mu(M'[\Uparrow^n (N/)]) = \mu(M[S]\ M'[S])$.

[5] In [BBLRD95] notation, the v-system is obtained by dropping the rule (Beta) from the rewriting system λ_v.

– $S = \Uparrow^n (\uparrow)$. We have that $\mu((M \odot M')[S]) = \tau_n^1(\mu(M \odot M')) = \tau_n^1(\mu(M)\ \mu(M')) = \tau_n^1(\mu(M))\ \tau_n^1(\mu(M')) = \mu(M[\Uparrow^n (\uparrow)])\ \mu(M'[\Uparrow^n (\uparrow)]) = \mu(M[S]\ M'[S])$.

- Case (Mark). In order to prove that $\mu((MM')[S]) = \mu((M \bullet M')[S])$, we consider two cases.

  – $S = \Uparrow^n (N/)$. We have that $\mu((MM')[S]) = \sigma_n(\mu(MM'), \mu(N)) = \sigma_n(\mu(M)\ \mu(M'), \mu(N)) = \sigma_n(\mu(M \bullet M'), \mu(N)) = \mu((M \bullet M')[S])$.

  – $S = \Uparrow^n (\uparrow)$. We have that $\mu((MM')[S]) = \tau_n^1(\mu(MM')) = \tau_n^1(\mu(M)\ \mu(M')) = \tau_n^1(\mu(M \bullet M')) = \mu((M \bullet M')[S])$.

□

Corollary 2 *If M, N and S are ground λ_ζ-terms then $\mu((M \bullet N)[S]) = \mu((M \odot N)[S]) = \mu((M\ N)[S]) = \mu(M[S])\ \mu(N[S])$.*

An useful property of the function μ established for the system λ_v in [BBLRD95] say that $\mu(M[N/][S_1] \dots [S_n]) = \mu(M[\Uparrow (S_1)] \dots [\Uparrow (S_n)][N[S_1] \dots [S_n]/])$. We extend this result to system λ_ζ.

Lemma 9 *For any M, N and S ground λ_ζ-terms,*
$\mu(M[N/][S_1] \dots [S_n]) = \mu(M[\Uparrow (S_1)] \dots [\Uparrow (S_n)][N[S_1] \dots [S_n]/])$.

Proof. Let M',N' and $S'_1, \dots, S'_n$ be respectively the terms M,N and $S_1, \dots, S_n$, from which we have replaced all the marks by applications. By Lemma 8 and Corollary 2 we have that $\mu(M[N/][S_1] \dots [S_n]) = \mu(M'[N'/][S'_1] \dots [S'_n])$. Of course the term $M'[N'/][S'_1] \dots [S'_n]$ is a λ_v-term, then we can use the property stated above to obtain $\mu(M'[N'/][S'_1] \dots [S'_n]) = \mu(M'[\Uparrow (S'_1)] \dots [\Uparrow (S'_n)][N'[S'_1] \dots [S'_n]/])$. We use again Lemma 8 and Corollary 2 to conclude that $\mu(M'[\Uparrow (S'_1)] \dots [\Uparrow (S'_n)][N'[S'_1] \dots [S'_n]/]) =$
$\mu(M[\Uparrow (S_1)] \dots [\Uparrow (S_n)][N[S_1] \dots [S_n]/])$. □

Lemma 10 *Let M and N be ground λ_ζ-terms. If $M \xrightarrow{B^\bullet} N$ then $\mu(M) \xrightarrow{\beta} \mu(N)$.*

Proof. By structural induction on M, and cases analysis in the $B^\bullet$-rule applied to reduce M to N.

- If $M = (\lambda M')M"$ and $N = M'[M"/]$ with $M \xrightarrow{\text{(Beta)}} N$. By definition of μ, $\mu(M) = (\lambda\mu(M'))\mu(M")$ and $\mu(N) = \sigma_0(\mu(M'), \mu(M"))$, then by definition of β, $\mu(M) \xrightarrow{\beta} \mu(N)$.

- If $M = (\lambda M') \bullet M"$ and $N = M'[M"/]$ with $M \xrightarrow{\text{(Beta}^\bullet\text{)}} N$. By definition of μ, $\mu(M) = (\lambda\mu(M'))\mu(M")$ and $\mu(N) = \sigma_0(\mu(M'), \mu(M"))$, then by definition of β, $\mu(M) \xrightarrow{\beta} \mu(N)$.

□

Lemma 11 *Let M and N be ground λ_ζ-terms. If $M \xrightarrow{\lambda_\zeta^*} N$ then $\mu(M) \xrightarrow{\beta^*} \mu(N)$.*

Proof. We prove by induction on n that if $M \xrightarrow{\lambda_\zeta^n} N$ then $\mu(M) \xrightarrow{\beta^*} \mu(N)$. The base case if obvious. Assume that $M \xrightarrow{\lambda_\zeta} M' \xrightarrow{\lambda_\zeta^n} N$. If M is reduced to M' using a ζ-rule then, by Lemma 8, $\mu(M) = \mu(M')$ and, by the induction hypothesis, $\mu(M') \xrightarrow{\beta^*} \mu(N)$. Otherwise, if M is reduced to M' using a $B^\bullet$-rule then, by Lemma 10, $\mu(M) \xrightarrow{\beta} \mu(M')$ and, by the induction hypothesis, $\mu(M') \xrightarrow{\beta^*} \mu(N)$. □

Theorem 1 (λ_ζ-Soundness) *Let M be a λ_ζ-normalisable pure term, if $\widehat{M}$ is a λ_ζ-normal form of M, then $\widehat{M}$ is a β-normal form of M.*

Proof. M is a pure term then, by Corollary 1, $\widehat{M}$ is a pure term too, so $M = \mu(M)$ and $\widehat{M} = \mu(\widehat{M})$. By Lemma 11, $\mu(M) \xrightarrow{\beta^*} \mu(\widehat{M})$, therefore $M \xrightarrow{\beta^*} \widehat{M}$. Since the set of λ_ζ-normal form is included in the set of β-normal form, we conclude that $\widehat{M}$ is a β-normal form of M. □

4.2 λ_ζ-Completeness

The converse of the Soundness Theorem is also true, but its proof is a little more difficult because we consider a "big step" implementation of β-reduction.

First, we prove some natural properties of the system ζ.

Lemma 12 *The system ζ is terminating.*

Proof. It is not very difficult to check that the interpretation given in Fig. 4 defines a reduction order for ζ. □

Lemma 13 *The system ζ is confluent.*

Proof. We verify mechanically that the system ζ is locally confluent, for example using the system RRL ([KZ89]). ByLemma 12 ζ is terminating then using the Newman's lemma (Lemma 1) we conclude that ζ is confluent. □

Lemma 14 *The set of ζ-normal forms of purifiable terms is described by:*

ζ-Normal Form	$\widehat{P}$	$::=$	$\underline{n} \mid \lambda\widehat{P} \mid \widehat{P}\,\widehat{P'} \mid \widehat{H}[\widehat{S}]$
Header (Beta$^\bullet$)-redex	$\widehat{H}$	$::=$	$(\lambda\widehat{P}) \bullet \widehat{P'} \mid \mid \widehat{H} \bullet \widehat{P} \mid (\widehat{H} \bullet \widehat{P})[\widehat{S}]$
Substitution ζ-Normal Form	$\widehat{S}$	$::=$	$\widehat{P}/ \mid \uparrow \mid \Uparrow(\widehat{S})$

$[\![\underline{n}]\!]_1$	$=$	2^n	$[\![\underline{n}]\!]_2$	$=$	n
$[\![\lambda M]\!]_1$	$=$	$1 + [\![M]\!]_1$	$[\![\lambda M]\!]_2$	$=$	1
$[\![M\ M']\!]_1$	$=$	$1 + [\![M]\!]_1 + [\![M']\!]_1$	$[\![M\ M']\!]_2$	$=$	$3 + [\![M]\!]_2 + [\![M']\!]_2$
$[\![M \bullet M']\!]_1$	$=$	$1 + [\![M]\!]_1 + [\![M']\!]_1$	$[\![M \bullet M']\!]_2$	$=$	$2 + [\![M]\!]_2 + [\![M']\!]_2$
$[\![M \odot M']\!]_1$	$=$	$1 + [\![M]\!]_1 + [\![M']\!]_1$	$[\![M \odot M']\!]_2$	$=$	$1 + [\![M]\!]_2 + [\![M']\!]_2$
$[\![M[S]]\!]_1$	$=$	$[\![M]\!]_1 [\![S]\!]_1$	$[\![M[S]]\!]_2$	$=$	$[\![M]\!]_2 + [\![S]\!]_2$
$[\![M/]\!]_1$	$=$	$[\![M]\!]_1$	$[\![M/]\!]_2$	$=$	1
$[\![\uparrow]\!]_1$	$=$	2	$[\![\uparrow]\!]_2$	$=$	2
$[\![\Uparrow (S)]\!]_1$	$=$	$[\![S]\!]_1$	$[\![\Uparrow (S)]\!]_2$	$=$	$2 + [\![S]\!]_2$

Figure 4: Interpretation for proving the termination of ζ

Proof. Since $\zeta \subseteq \lambda_\zeta$, the Lemma 6 holds for ζ. Therefore, any ζ-normal form $\widehat{P}$ of a purifiable term is a purifiable term too. Now, we verify that terms of the form $\underline{n}[S]$, $(\lambda P)[S]$, $(P\ P')[S]$, $\overset{\odot}{P}[S]$, $\overset{\odot}{P} \bullet P'$ and $(P\ P') \bullet P$" are not ζ-normal forms. Thus the grammar above describe all the ζ-normal forms of purifiable terms. □

Definition 10 (Leftmost-Outermost Strategies)

- *Let M be a pure term, we say that M reduces to M' by a* Leftmost-Outermost β-strategy, *$M \xrightarrow[\text{LOS}]{\beta} M'$, if and only if $M \xrightarrow{\beta} M'$ using the leftmost-outermost β-redex of M.*
- *Let M be a ground λ_ζ-term, we say that M reduces to M' by a* Leftmost-Outermost $B^\bullet$-strategy, *$M \xrightarrow[\text{LOS}]{B^\bullet} M'$, if and only if $M \xrightarrow{B^\bullet} M'$ using the leftmost-outermost $B^\bullet$-redex of M.*
- *Let M be a ground λ_ζ-term, we say that M reduces to $\widehat{N}$ by a* Leftmost-Outermost λ_ζ-strategy, *$M \xrightarrow[\text{LOS}]{\lambda_\zeta} \widehat{N}$, if and only if $M \xrightarrow[\text{LOS}]{B^\bullet} N \xrightarrow{\zeta^*} \widehat{N}$ and $\widehat{N}$ is the ζ-normal form of N.*

The following lemmas establish the relation between leftmost-outermost λ_ζ and β-strategies.

Lemma 15 *Let $\widehat{H}$ be a header (Beta$^\bullet$)-redex (as defined in Lemma 14). There exists a term Q such that (1) $\widehat{H} \xrightarrow[\text{LOS}]{B^\bullet} Q$, and (2) for any $S_1, S_2, \ldots, S_n$ ground substitutions, $\mu(\widehat{H}[S_1] \ldots [S_n]) \xrightarrow[\text{LOS}]{\beta} \mu(Q[S_1] \ldots [S_n])$.*

Proof. By structural induction on $\widehat{H}$.

- If $\widehat{H} = (\lambda\widehat{P}) \bullet \widehat{P'}$ then we take $Q = \widehat{P}[\widehat{P'}/]$ and using (Beta$^\bullet$) (1) $(\lambda\widehat{P}) \bullet \widehat{P'} \xrightarrow[\text{LOS}]{B^\bullet} Q$. (2) By Lemma 8 and Corollary 2 we have that $\mu(\widehat{H}[S_1]\dots[S_n]) = (\lambda\mu(\widehat{P}[\Uparrow(S_1)]\dots[\Uparrow(S_n)]))\ \mu(\widehat{P'}[S_1]\dots[S_n])$. Thus, by definition of leftmost-outermost β-strategy we have that $\mu(\widehat{H}[S_1]\dots[S_n]) \xrightarrow[\text{LOS}]{\beta} \sigma_0(\mu(\widehat{P}[\Uparrow(S_1)]\dots[\Uparrow(S_n)]), \mu(\widehat{P'}[S_1]\dots[S_n]))$. Using definition of μ we have that $\sigma_0(\mu(\widehat{P}[\Uparrow(S_1)]\dots[\Uparrow(S_n)]), \mu(\widehat{P'}[S_1]\dots[S_n])) = \mu(\widehat{P}[\Uparrow(S_1)]\dots[\Uparrow(S_n)][\widehat{P'}[S_1]\dots[S_n]/])$. By Lemma 9 we have that $\mu(\widehat{P}[\Uparrow(S_1)]\dots[\Uparrow(S_n)][\widehat{P'}[S_1]\dots[S_n]/]) = \mu(\widehat{P}[\widehat{P'}/][S_1]\dots[S_n])$, but also $\mu(Q[S_1]\dots[S_n]) = \mu(\widehat{P}[\widehat{P'}/][S_1]\dots[S_n])$.
- If $\widehat{H} = \widehat{H'} \bullet \widehat{P}$ then by induction hypothesis $\widehat{H'} \xrightarrow[\text{LOS}]{B^\bullet} Q'$. We take $Q = Q' \bullet \widehat{P}$ and we have (1) $\widehat{H'} \bullet \widehat{P} \xrightarrow[\text{LOS}]{B^\bullet} Q' \bullet \widehat{P}$. (2) By Lemma 8 and Corollary 2 we have that $\mu(\widehat{H}[S_1]\dots[S_n]) = \mu(\widehat{H'}[S_1]\dots[S_n])\ \mu(\widehat{P}[S_1]\dots[S_n])$. By induction hypothesis, $\mu(\widehat{H'}[S_1]\dots[S_n]) \xrightarrow[\text{LOS}]{\beta} \mu(Q'[S_1]\dots[S_n])$, thus we have also $\mu(\widehat{H}[S_1]\dots[S_n]) \xrightarrow[\text{LOS}]{\beta} \mu(Q'[S_1]\dots[S_n])\ \mu(\widehat{P}[S_1]\dots[S_n])$. We conclude using again Lemma 8 and Corollary 2 that $\mu(Q[S_1]\dots[S_n])\ \mu(\widehat{P}[S_1]\dots[S_n]) = \mu((Q'\bullet P)[S_1]\dots[S_n])$.
- If $\widehat{H} = (\widehat{H'} \bullet \widehat{P})[\widehat{S}]$ then by induction hypothesis $\widehat{H'} \xrightarrow[\text{LOS}]{B^\bullet} Q'$. We take $Q = (Q' \bullet \widehat{P})[\widehat{S}]$ and we have (1) $(\widehat{H'} \bullet \widehat{P})[\widehat{S}] \xrightarrow[\text{LOS}]{B^\bullet} (Q' \bullet \widehat{P})[\widehat{S}]$. (2) By Lemma 8 and Corollary 2 we have that $\mu(\widehat{H}[S_1]\dots[S_n]) = \mu(\widehat{H'}[\widehat{S}][S_1]\dots[S_n])\ \mu(\widehat{P}[\widehat{S}][S_1]\dots[S_n])$. By induction hypothesis, $\mu(\widehat{H'}[\widehat{S}][S_1]\dots[S_n]) \xrightarrow[\text{LOS}]{\beta} \mu(Q'[\widehat{S}][S_1]\dots[S_n])$, thus we have also $\mu(\widehat{H}[S_1]\dots[S_n]) \xrightarrow[\text{LOS}]{\beta} \mu(Q'[\widehat{S}][S_1]\dots[S_n])\ \mu(\widehat{P}[\widehat{S}][S_1]\dots[S_n])$. We conclude using again Lemma 8 and Corollary 2 that $\mu(Q[\widehat{S}][S_1]\dots[S_n])\ \mu(\widehat{P}[\widehat{S}][S_1]\dots[S_n]) = \mu((Q' \bullet P)[\widehat{S}][S_1]\dots[S_n])$.

□

Lemma 16 *Let $\widehat{P}$ be a ζ-normal form of a purifiable term. If $\widehat{P} \xrightarrow[\text{LOS}]{B^\bullet} Q$ then $\mu(\widehat{P}) \xrightarrow[\text{LOS}]{\beta} \mu(Q)$.*

Proof. By structural induction on $\widehat{P}$.

- If $\widehat{P} = \underline{n}$ then $\widehat{P}$ is not $B^\bullet$-reducible, and then the result is obvious.
- If $\widehat{P} = \lambda\widehat{P'}$ and $\lambda\widehat{P} \xrightarrow[\text{LOS}]{B^\bullet} Q$, then $Q = \lambda Q'$ and $\widehat{P'} \xrightarrow[\text{LOS}]{B^\bullet} Q'$. By definition of μ, $\mu(\widehat{P}) = \lambda\mu(\widehat{P'})$ and $\mu(Q) = \lambda\mu(Q')$. By induction hypothesis $\mu(\widehat{P'}) \xrightarrow[\text{LOS}]{\beta} \mu(Q')$. Thus, we conclude that $\mu(\widehat{P}) \xrightarrow[\text{LOS}]{\beta} \mu(Q)$.

- If $\widehat{P} = (\lambda\widehat{P'})\widehat{P''}$ then $\overbrace{(\lambda\widehat{P'})\widehat{P''}}^{B^\bullet\text{-redex}} \xrightarrow[\text{LOS}]{B^\bullet} \widehat{P'}[\widehat{P''}/]$. By definition of μ we have that $\mu(\widehat{P}) = (\lambda\mu(\widehat{P'}))\mu(\widehat{P''})$ and $\mu(\widehat{P'}[\widehat{P''}/]) = \sigma_0(\mu(\widehat{P'}), \mu(\widehat{P''}))$, but also $\overbrace{(\lambda\mu(\widehat{P'}))\mu(\widehat{P''})}^{\beta\text{-redex}} \xrightarrow[\text{LOS}]{\beta} \sigma_0(\mu(\widehat{P'}), \mu(\widehat{P''}))$.

- If $\widehat{P} = \widehat{P'}\ \widehat{P''}$, where $\widehat{P'}$ is not a λ abstraction, and $\widehat{P} \xrightarrow[\text{LOS}]{B^\bullet} Q$, then there are two cases:
  - $Q = Q'\ \widehat{P''}$ and $\widehat{P'} \xrightarrow[\text{LOS}]{B^\bullet} Q'$. By definition of μ, $\mu(\widehat{P}) = \mu(\widehat{P'})\ \mu(\widehat{P''})$ and $\mu(Q) = \mu(Q')\ \mu(\widehat{P''})$. By induction hypothesis $\mu(\widehat{P'}) \xrightarrow[\text{LOS}]{\beta} \mu(Q')$. Thus, we conclude that $\mu(\widehat{P}) \xrightarrow[\text{LOS}]{\beta} \mu(Q)$.
  - $Q = \widehat{P'}\ Q''$ and $\widehat{P''} \xrightarrow[\text{LOS}]{B^\bullet} Q''$. By definition of μ, $\mu(\widehat{P}) = \mu(\widehat{P'})\ \mu(\widehat{P''})$ and $\mu(Q) = \mu(\widehat{P'})\ \mu(Q'')$. By induction hypothesis $\mu(\widehat{P''}) \xrightarrow[\text{LOS}]{\beta} \mu(Q'')$. Thus, we conclude that $\mu(\widehat{P}) \xrightarrow[\text{LOS}]{\beta} \mu(Q)$.

- If $\widehat{P} = \widehat{H}[\widehat{S}]$, then by Lemma 15, there is a Q such that $\widehat{H} \xrightarrow[\text{LOS}]{B^\bullet} Q$ and $\mu(\widehat{H}[\widehat{S}]) \xrightarrow[\text{LOS}]{\beta} \mu(Q[\widehat{S}])$.

□

Notice that the previous lemma is not valid if $\widehat{P}$ is not a ζ-normal form, for example $(\underline{1}\ (\overbrace{(\lambda\underline{1})\underline{2}}^{B^\bullet\text{-redex}}))[\lambda\underline{2}/] \xrightarrow[\text{LOS}]{B^\bullet} (\underline{1}\ \underline{1}[\underline{2}/])[\lambda\underline{2}/]$, but $\mu((\underline{1}\ ((\lambda\underline{1})\underline{2}))[\lambda\underline{2}/]) = (\lambda\underline{2})\ ((\lambda\underline{1})\underline{1})$ and $\overbrace{(\lambda\underline{2})\ ((\lambda\underline{1})\underline{1})}^{\beta\text{-redex}} \xrightarrow[\text{LOS}]{\beta} \underline{1}$. Of course $\mu((\underline{1}\ \underline{1}[\underline{2}/])[\lambda\underline{2}/]) \neq \underline{1}$.

Lemma 17 *Let $\widehat{P}$ be a ζ-normal form of a purifiable term, if $\widehat{P} \xrightarrow[\text{LOS}]{\lambda_\zeta} \widehat{Q}$ then $\mu(\widehat{P}) \xrightarrow[\text{LOS}]{\beta} \mu(\widehat{Q})$.*

Proof. By definition of leftmost-outermost λ_ζ-strategy, $\widehat{P} \xrightarrow[\text{LOS}]{B^\bullet} Q \xrightarrow{\zeta^*} \widehat{Q}$ and $\widehat{Q}$ is the ζ-normal form of Q. By Lemma 16, $\mu(\widehat{P}) \xrightarrow[\text{LOS}]{\beta} \mu(Q)$ and by Lemma 8 we conclude that $\mu(Q) = \mu(\widehat{Q})$. □

Lemma 18 *The Leftmost-Outermost β-strategy is a normal strategy, i.e. if M is a β-normalisable term then there exists a β-normal form $\widehat{M}$ such that $M \xrightarrow[\text{LOS}]{\beta^*} \widehat{M}$.*

Proof. See [Bar84] □

Theorem 2 (λ_ζ-Completeness) *Let M be a β-normalisable term, if $\widehat{M}$ is the β-normal form of M, then $\widehat{M}$ is a λ_ζ-normal form of M.*

Proof. First, we remark that: β-Normal Terms $\subseteq$ β-Terms = Pure Terms $\subseteq$ ζ-Normal Forms of Purifiable Terms $\subseteq$ Purifiable Terms.

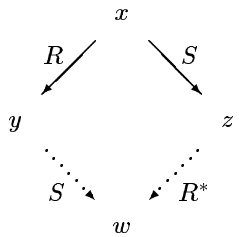
Let M be a β-normalisable term, assume that there exists an infinite reduction $M = M_0 \xrightarrow[\text{LOS}]{\lambda_\zeta} M_1 \xrightarrow[\text{LOS}]{\lambda_\zeta} M_2 \xrightarrow[\text{LOS}]{\lambda_\zeta} \ldots$. By the above remark and Lemma 6, all the $M_i (i \leq 0)$ are purifiable terms. By definition of leftmost-outermost λ_ζ-reduction, we have that all the M_i are ζ-normal forms (of purifiable terms). Using Lemma 17, we have that $\mu(M) \xrightarrow[\text{LOS}]{\beta} \mu(M_1) \xrightarrow[\text{LOS}]{\beta} \mu(M_2) \xrightarrow[\text{LOS}]{\lambda_\zeta} \ldots$ is an infinite reduction too. But, M is a β-term then $\mu(M) = M$, so there is an infinite leftmost-outermost β-reduction of M and this is contradictory with Lemma 18. Therefore there is a reduction $M \xrightarrow[\text{LOS}]{\lambda_\zeta *} \widehat{M'}$ and $\widehat{M'}$ is a λ_ζ-normal form (of a purifiable term). By Theorem 1, $\widehat{M'}$ is a β-normal form too, and because λ-calculus is confluent the normal form is unique (Lemma 3), so $\widehat{M'} = \widehat{M}$. □

5 Confluence

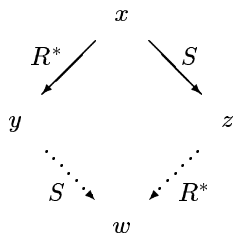
Proposition 3 (Confluence) *λ_ζ is confluent.*

We address now to search an abstract sufficient condition to prove λ_ζ-confluence. This development is similar to [YH88] and [CHL95].

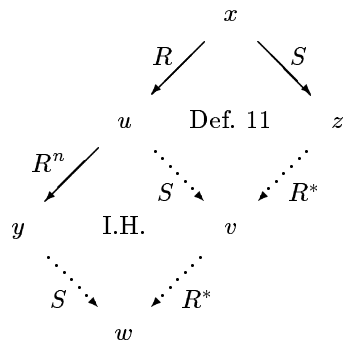
Definition 11 (Strong Closure) *Let S and R be two relations defined on the set X. We say that R is* strongly closed on S *if and only if for any x, y, z in X such that $x \xrightarrow{R} y$ and $x \xrightarrow{S} z$, there exists w in X such that $y \xrightarrow{S} w$ and $z \xrightarrow{R^*} w$, i.e. the following diagram holds:*



Lemma 19 *Let S and R be two relations such that R is strongly closed on S. For any x, y, z in X such that $x \xrightarrow{R^*} y$ and $x \xrightarrow{S} z$, there exists w in X such that $y \xrightarrow{S} w$ and $z \xrightarrow{R^*} w$, i.e. the following diagram holds:*



Proof. By induction on the R-depth $x \xrightarrow{R^*} y$. If $x = y$ then the conclusion is trivial taking $z = w$. In the induction step we assume that the property holds for R^n and that $x \xrightarrow{R^{n+1}} y$ and $x \xrightarrow{S} z$. We take u such that $x \xrightarrow{R} u \xrightarrow{R^n} y$. Since R is strongly closed on S, there is v such that $u \xrightarrow{S} v$ and $z \xrightarrow{R^*} v$. By induction hypothesis we conclude that there exists w such that $y \xrightarrow{S} w$ and $v \xrightarrow{R^*} w$. Graphically:

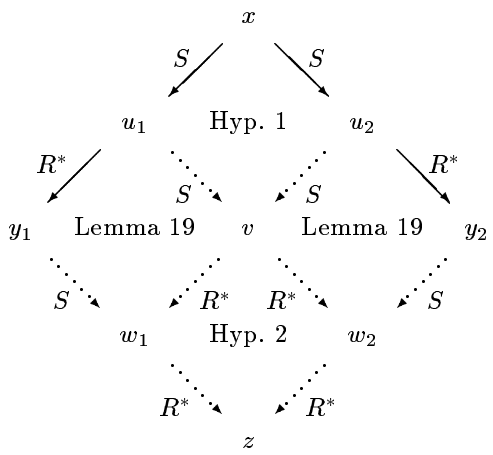


□

Lemma 20 *Let R and S two relations such that (1) S is strongly confluent and (2) R confluent. If R is strongly closed on S then the relation SR^* is strongly confluent (and therefore, by Lemma 2, confluent).*

Proof. Let x be such that $x \xrightarrow{SR^*} y_1$ and $x \xrightarrow{SR^*} y_2$. We assume that there are u_1,u_2 such that $x \xrightarrow{S} u_1 \xrightarrow{R^*} y_1$ and $x \xrightarrow{S} u_2 \xrightarrow{R^*} y_2$. Since S is strongly confluent, there is v such that $u_1 \xrightarrow{S} v$ and $u_2 \xrightarrow{S} v$. By hypothesis R is strongly closed on S, then using Lemma 19 there are w_1 and w_2 such that $y_1 \xrightarrow{S} w_1$, $v \xrightarrow{R^*} w_1$, $v \xrightarrow{R^*} w_2$ and $y_2 \xrightarrow{S} w_2$. Finally, by hypothesis (2) R is confluent, then there is z such that $w_1 \xrightarrow{R^*} z$ and $w_2 \xrightarrow{R^*} z$.

Graphically:



□

We shall apply the last lemma taking X as the set of λ_ζ-terms, S as the relation induced by the parallelisation of $B^\bullet$ (Def. 8), and R as the relation induced by the system ζ (Def. 8). On this way, we present the following definition and lemma:

Definition 12 *A term is* linear *if and only if every variable that it contains occurs only once. A rewriting system is* left linear *(respectively,* right linear*) if and only if for any system's rule $l \longrightarrow r$, l (respectively, r) is linear.*

Lemma 21 *Let $R_\parallel$ be the parallel reduction by rules on R at disjoint occurrences. If R is left linear with no critical pairs then (1) the relation $R_\parallel$ is strongly confluent and (2) the relation R is confluent.*

Proof. See [Hue80]. □

Now, we prove the hypothesis (1) of Lemma 20. The parallelisation of $B^\bullet$ ($B^\bullet_\parallel$) is defined by:

$$\frac{}{M \longrightarrow M}\,(\text{Refl-Term}_\parallel) \qquad \frac{}{S \longrightarrow S}\,(\text{Refl-Subs}_\parallel)$$

$$\frac{M \longrightarrow M'}{\lambda M \longrightarrow \lambda M'}\,(\text{Lbd}_\parallel) \qquad \frac{M \longrightarrow M'}{M/ \longrightarrow M'/}\,(\text{Sl}_\parallel)$$

$$\frac{M \longrightarrow M' \qquad S \longrightarrow S'}{M[S] \longrightarrow M'[S']}\,(\text{Subs}_\parallel) \qquad \frac{S \longrightarrow S'}{\Uparrow(S) \longrightarrow \Uparrow(S')}\,(\text{Lft}_\parallel)$$

$$\frac{M \longrightarrow M' \qquad N \longrightarrow N'}{M\ N \longrightarrow M'\ N'}\,(\text{Application}_\parallel) \qquad \frac{M \longrightarrow M' \qquad N \longrightarrow N'}{M \bullet N \longrightarrow M' \bullet N'}\,(\text{Application}^\bullet_\parallel)$$

$$\frac{M \longrightarrow M' \qquad N \longrightarrow N'}{M \odot N \longrightarrow M' \odot N'}\,(\text{Application}^\odot_\parallel) \qquad \frac{M \longrightarrow M' \qquad N \longrightarrow N'}{(\lambda M)N \longrightarrow M'[N'/]}\,(\text{Beta}_\parallel)$$

$$\frac{M \longrightarrow M' \qquad N \longrightarrow N'}{(\lambda M) \bullet N \longrightarrow M'[N'/]}\,(\text{Beta}^\bullet_\parallel)$$

Lemma 22 *The rewriting system $B^\bullet_\parallel$ is strongly confluent.*

Proof. The rewriting system $B^\bullet$ is left linear and we check easily that it does not contain critical pairs, then by Lemma 21 the rewriting system $B^\bullet_\parallel$ is strongly confluent. □

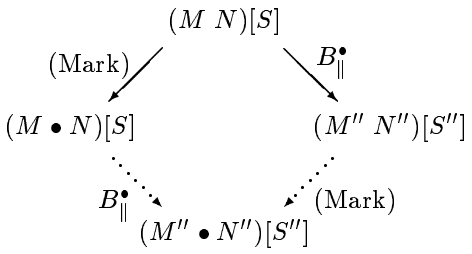
The hypothesis (2) of Lemma 20 holds, because the rewriting system ζ is confluent (Lemma 13). Finally, we addresses the last condition of Lemma 20.

Lemma 23 *The relation induced by ζ is strongly closed on $B^\bullet_\parallel$.*

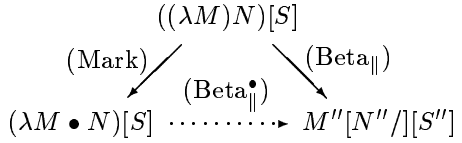
Proof. Assume that an arbitrary term T reduces in one step to T' with ζ and to T'' with $B^\bullet_\parallel$. We proceed by induction on the depth of the ζ-redex reduced in T.

- At the base case T is a ζ-redex. We work by cases on the ζ-rule applied to reduce T.
  - Case (Mark). There are two cases:
    - Assume $T = (M\ N)[S]$, $T' = (M \bullet N)[S]$ and $T'' = (M''\ N'')[S'']$ with $T \xrightarrow{\text{(Mark)}} T'$, $M \xrightarrow{B^\bullet_\parallel} M''$, $N \xrightarrow{B^\bullet_\parallel} N''$ and $S \xrightarrow{B^\bullet_\parallel} S''$. Then by definition of $B^\bullet_\parallel$: $(M \bullet N)[S] \xrightarrow{B^\bullet_\parallel} (M'' \bullet N'')[S'']$. But also

$(M''\ N'')[S'] \xrightarrow{\text{(Mark)}} (M'' \bullet N'')[S']$. Graphically,

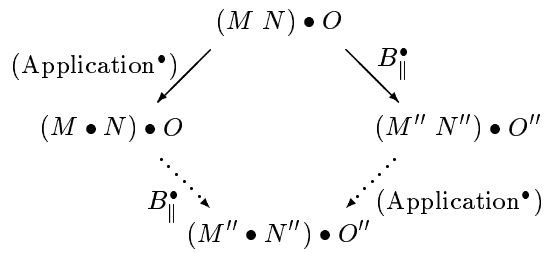


* Assume $T = ((\lambda M)N)[S]$, $T' = (\lambda M \bullet N)[S]$ and $T'' = M''[N''/][S'']$ with $T \xrightarrow{\text{(Mark)}} T'$, $M \xrightarrow{B^\bullet_\parallel} M''$, $N \xrightarrow{B^\bullet_\parallel} N''$, $S \xrightarrow{B^\bullet_\parallel} S''$ and $T \xrightarrow{(\text{Beta}_\parallel)} T''$. Also we have that $(\lambda M \bullet N)[S] \xrightarrow{(\text{Beta}^\bullet_\parallel)} M''[N''/][S'']$. Graphically,



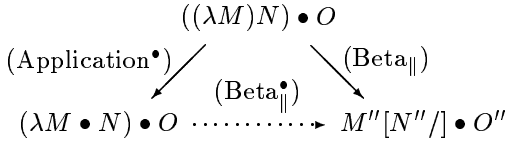
– Case (Application•). There are two cases:

* Assume $T = (M\ N) \bullet O$, $T' = (M \bullet N) \bullet O$ and $T'' = (M''\ N'') \bullet O''$ with $T \xrightarrow{(\text{Application}^\bullet)} T'$, $M \xrightarrow{B^\bullet_\parallel} M''$, $N \xrightarrow{B^\bullet_\parallel} N''$ and $O \xrightarrow{B^\bullet_\parallel} O''$. Then by definition of $B^\bullet_\parallel$: $(M \bullet N) \bullet O \xrightarrow{B^\bullet_\parallel} (M'' \bullet N'') \bullet O''$. But also $(M''\ N'') \bullet O'' \xrightarrow{(\text{Application}^\bullet)} (M'' \bullet N'') \bullet O''$. Graphically,

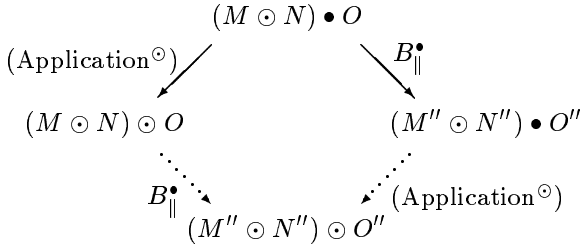


* Assume $T = ((\lambda M)N) \bullet O$, $T' = (\lambda M \bullet N) \bullet O$ and $T'' = M''[N''/] \bullet O''$ with $T \xrightarrow{(\text{Application}^\bullet)} T'$, $M \xrightarrow{B^\bullet_\parallel} M''$, $N \xrightarrow{B^\bullet_\parallel} N''$, $O \xrightarrow{B^\bullet_\parallel} O''$ and $T \xrightarrow{(\text{Beta}_\parallel)} T''$.

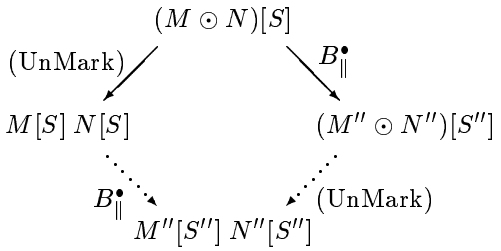
Also we have that $(\lambda M \bullet N) \bullet O \xrightarrow{(\mathrm{Beta}^{\bullet}_{\parallel})} M''[N''/] \bullet O''$. Graphically,



– Case (Application$^{\odot}$). Assume $T = (M \odot N) \bullet O$, $T' = (M \odot N) \odot O$ and $T'' = (M'' \odot N'') \bullet O''$ with $T \xrightarrow{(\mathrm{Application}^{\odot})} T'$, $M \xrightarrow{B^{\bullet}_{\parallel}} M''$, $N \xrightarrow{B^{\bullet}_{\parallel}} N''$ and $O \xrightarrow{B^{\bullet}_{\parallel}} O''$. Then by definition of $B^{\bullet}_{\parallel}$: $(M \odot N) \odot O \xrightarrow{B^{\bullet}_{\parallel}} (M'' \odot N'') \odot O''$.
But also $(M'' \odot N'') \bullet O'' \xrightarrow{(\mathrm{Application}^{\odot})} (M'' \odot N'') \odot O''$. Graphically,

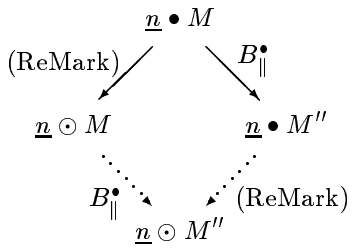


– Case (UnMark). Assume $T = (M \odot N)[S]$, $T' = M[S]\ N[S]$ and $T'' = (M'' \odot N'')[S'']$ with $T \xrightarrow{(\mathrm{UnMark})} T'$, $M \xrightarrow{B^{\bullet}_{\parallel}} M''$, $N \xrightarrow{B^{\bullet}_{\parallel}} N''$ and $S \xrightarrow{B^{\bullet}_{\parallel}} S''$. Then by definition of $B^{\bullet}_{\parallel}$: $M[S]\ N[S] \xrightarrow{B^{\bullet}_{\parallel}} M''[S'']\ N''[S'']$.
But also $(M'' \odot N'')[S''] \xrightarrow{(\mathrm{UnMark})} M''[S'']\ N''[S'']$. Graphically,



– Case (ReMark). Assume $T = \underline{n} \bullet M$, $T' = \underline{n} \odot M$ and $T'' = \underline{n} \bullet M''$ with $T \xrightarrow{(\mathrm{ReMark})} T'$, $M \xrightarrow{B^{\bullet}_{\parallel}} M''$. Then by definition of $B^{\bullet}_{\parallel}$: $\underline{n} \odot M \xrightarrow{B^{\bullet}_{\parallel}} \underline{n} \odot M''$.

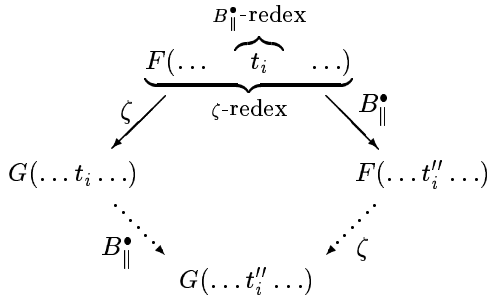
But also $\underline{n} \bullet M'' \xrightarrow{\text{(ReMark)}} \underline{n} \odot M''$. Graphically,



– The rule ζ is one of (Lambda), (FVar), (RVar), (FVarLift), (RVarLift) and (VarShift). Notice that:

1. All these ζ-rules are left and right linear.
2. A strict subexpression of these ζ-redexes can never overlap with a $B^{\bullet}_{\|}$-redex.
3. There are no common redexes between these ζ-rules and $B^{\bullet}_{\|}$-rules.

Without lost of generality we assume $T = F(\dots t_i \dots)$, $T' = G(\dots t_i \dots)$ and $T'' = F(\dots t''_i \dots)$ with $t_i \xrightarrow{B^{\bullet}_{\|}} t''_i$, then we have that $T' \xrightarrow{B^{\bullet}_{\|}} G(\dots t''_i \dots)$. By remarks (a) and (b) the same ζ-redex of T is kept in T'' and therefore we can conclude that $T'' \xrightarrow{\zeta} G(\dots t''_i \dots)$. Graphically,

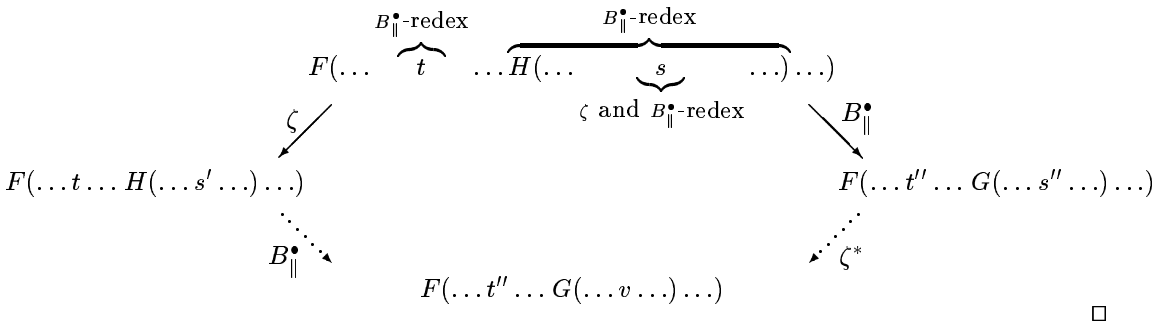


- At the induction step we assume, without lost of generality, that $T = F(\dots t \dots H(\dots s \dots) \dots)$, $T' = F(\dots t \dots H(\dots s' \dots) \dots)$ and $T'' = F(\dots t'' \dots G(\dots s'' \dots) \dots)$ with $s \xrightarrow{\zeta} s'$, $t \xrightarrow{B^{\bullet}_{\|}} t''$, $s \xrightarrow{B^{\bullet}_{\|}} s''$ and $H(\dots s \dots) \xrightarrow{B^{\bullet}_{\|}} G(\dots s'' \dots)$. Notice that:

1. $B^{\bullet}_{\|}$ is left linear.
2. $B^{\bullet}_{\|}$ is right linear, then s is reduced to s'' at most only once.
3. A strict subexpression of a $B^{\bullet}_{\|}$-redex can never overlaps with a ζ-redex.

By induction hypothesis there exists v such that $s' \xrightarrow{B^{\bullet}_{\|}} v$ and $s'' \xrightarrow{\zeta^*} v$, then we have that $T'' \xrightarrow{\zeta^*} F(\dots t'' \dots G(\dots v \dots) \dots)$. Finally, by the previous remarks we

conclude that the $B^\bullet_{\|}$-redex of $H(\dots s \dots)$ is kept in $H(\dots s' \dots)$ and therefore we have that $T' \xrightarrow{B^\bullet_{\|}} F(\dots t'' \dots G(\dots v \dots) \dots)$. Graphically,



□

Theorem 3 (Confluence) *The system λ_ζ is confluent.*

Proof. Notice:

1. The system $B^\bullet_{\|}\ \zeta^*$ is confluent by Lemma 20 using Lemma 22, Lemma 13 and Lemma 23 as hypothesis.
2. $\lambda_\zeta \subseteq B^\bullet_{\|}\ \zeta^* \subseteq \lambda^*_\zeta$, by definition of systems.

Let x, y, z be such that $x \xrightarrow{\lambda^*_\zeta} y$ and $x \xrightarrow{\lambda^*_\zeta} z$. By remark (2) $x \xrightarrow{(B^\bullet_{\|}\ \zeta^*)^*} y$ and $x \xrightarrow{(B^\bullet_{\|}\ \zeta^*)^*} z$. Then, by remark (1) there exists w such that $y \xrightarrow{(B^\bullet_{\|}\ \zeta^*)^*} w$ and $z \xrightarrow{(B^\bullet_{\|}\ \zeta^*)^*} w$. Using again the remark (2) we can conclude that $y \xrightarrow{\lambda^*_\zeta} w$ and $z \xrightarrow{\lambda^*_\zeta} w$. This prove the confluence of λ_ζ. □

Using Lemma 3 we have

Corollary 3 *The λ_ζ-normal form of an element, if it exists, is unique.*

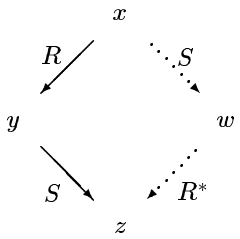
6 Preservation of strong normalisation

Proposition 4 (Preservation of Strong Normalisation) *If M is a strongly β-normalisable term, then M is a strongly λ_ζ-normalisable term.*

On the same way that we have worked with λ_ζ-confluence, we will search an abstract sufficient condition to prove that λ_ζ preserves strong normalisation.

Definition 13 (Commutation) *Let R and S be two relations defined on the set X. We say that R* commutes over *S if and only if for any x, y, z in X such that $x \xrightarrow{R} y$ and*

$y \xrightarrow{S} z$ *there exists* w *in* X *such that* $x \xrightarrow{S} w$ *and* $w \xrightarrow{R^*} z$, *i.e. the following diagram holds:*



In the following lemmas we assume that R, S are relations defined on the set X.

Lemma 24 *Let* R *and* S *be two relations such that* R *commutes over* S. *If there are* x, y *in* X *such that* $x \xrightarrow{R^*} \xrightarrow{S} y$ *then* $x \xrightarrow{S} \xrightarrow{R^*} y$.

Proof. We prove by induction on n that if $x \xrightarrow{R^n} \xrightarrow{S} y$, then $x \xrightarrow{S} \xrightarrow{R^*} y$. The base case is trivial. Assume that $x \xrightarrow{R^n} u \xrightarrow{R} v \xrightarrow{S} y$, then because R commutes over S we can deduce that there exists v' such that $x \xrightarrow{R^n} u \xrightarrow{S} v' \xrightarrow{R^*} y$. By induction hypothesis we conclude that $x \xrightarrow{S} \xrightarrow{R^*} v' \xrightarrow{R^*} y$, which means that $x \xrightarrow{S} \xrightarrow{R^*} y$. □

Lemma 25 *Let* R *and* S *be two relations such that (1)* R *is terminating and (2)* R *commutes over* S, *for any* x *in* X, *if* x *is strongly* S*-normalisable then* x *is strongly* $(R \cup S)$*-normalisable.*

Proof. Let x be in X such that x is strongly S-normalisable. We define the set of successors of x in S as $X_x = \{y \in X / x \xrightarrow{S^*} y\}$ and the order relation $(\succ)$ over X_x as: for all y, z in X_x, $y \succ z$ if and only if $y \xrightarrow{S^+} z$. Since x is strongly S-normalisable, the order $\succ$ is a well founded relation. Thus, we prove by noetherian induction over $\succ$ that for any y in X_x, y is strongly $(R \cup S)$-normalisable.

- At the base case, y is a S-normal form. Assume that there is a reduction $y \xrightarrow{(R \cup S)^*} \ldots$, then if there are not S-steps in the reduction, it has the form $y \xrightarrow{R^*} \ldots$ and this reduction is terminating by hypothesis (1). Otherwise, there is at most one S-step and we can write the reduction as $y \xrightarrow{R^*} \xrightarrow{S} z \xrightarrow{(R \cup S)^*} \ldots$. In this case we use the commutation hypothesis and Lemma 24 to show that there exists a reduction $y \xrightarrow{S} \xrightarrow{R^*} z \xrightarrow{(R \cup S)^*}$ and this is contradictory because y is a S-normal form. Therefore this last case does not apply.

- At the induction step assume that there is a reduction $y \xrightarrow{(R \cup S)^*} \ldots$, then if there are not S-steps in the reduction, it has the form $y \xrightarrow{R^*} \ldots$ and this reduction is terminating by hypothesis (1). Otherwise, there is at most one S-step and we can write the

$(M\ M')[S]$	$\longrightarrow$	$(M \bullet M')[S]$	(Mark)
$\underline{n} \bullet M$	$\longrightarrow$	$\underline{n} \odot M$	(ReMark)
$(M\ N) \bullet M'$	$\longrightarrow$	$(M \bullet N) \bullet M'$	(Application$^\bullet$)
$(M \odot N) \bullet M'$	$\longrightarrow$	$(M \odot N) \odot M'$	(Application$^\odot$)

Figure 5: The rewriting system $\mathcal{M}$

reduction as $y \xrightarrow{R^*} \xrightarrow{S} z \xrightarrow{(R\cup S)^*} \ldots$. In this case we use the commutation hypothesis and Lemma 24 to show that there exists a reduction $y \xrightarrow{S} w \xrightarrow{R^*} z \xrightarrow{(R\cup S)^*}$. By definition w is in X_x and $y \succ w$, then we can apply the induction hypothesis to w, to show that w is $(R \cup S)$-normalisable. Therefore z is $(R \cup S)$-normalisable too and we can conclude that the reduction $y \xrightarrow{R^*} \xrightarrow{S} z \xrightarrow{(R\cup S)^*} \ldots$ is finite.

Finally, by definition $x \in X_x$, thus x is strongly $(R \cup S)$-normalisable. □

In order to show that λ_ζ preserves strong normalisation, we shall apply the last lemma taking X as the set of λ_ζ-terms, R as the relation induced by the system $\mathcal{M}$ (Def. 8 and Fig. 5) and S as the relation induced by the system $\lambda_v^\bullet$ (Def. 14 and Fig. 6).

First, we prove the hypothesis (1) of Lemma 25.

Lemma 26 *The system $\mathcal{M}$ is terminating.*

Proof. Notice that $\mathcal{M} \subseteq \zeta$ and, by Lemma Lemma 12, ζ is terminating, then $\mathcal{M}$ is terminating too. □

Definition 14 *The system $\lambda_v^\bullet$ is defined as the system λ_v in addition with the rules:* (Beta$^\bullet$), (UnMark) *and:*

$$(M \bullet M')[S] \longrightarrow M[S]\ M'[S] \quad \text{(UnMark')}$$

Now, we address to the hypothesis (2) of Lemma 25.

Lemma 27 *The relation induced by the system $\mathcal{M}$ commutes over the relation induced by the system $\lambda_v^\bullet$.*

Proof. Assume that an arbitrary term T reduces in one step to T' with $\mathcal{M}$ and T' reduces in one step to T'' with $\lambda_v^\bullet$. We proceed by induction on the depth of the $\mathcal{M}$-redex reduced in T.

- At the base case T is a $\mathcal{M}$-redex. We work by cases on the $\mathcal{M}$ rule applied to reduce T.

$(\lambda M)M'$	$\longrightarrow$	$M[M'/]$	(Beta)
$(\lambda M) \bullet M'$	$\longrightarrow$	$M[M'/]$	(Beta$^\bullet$)
$(MM')[S]$	$\longrightarrow$	$M[S]\ M'[S]$	(Application)
$(\lambda M)[S]$	$\longrightarrow$	$\lambda(M[\Uparrow (S)])$	(Lambda)
$\underline{1}[M/]$	$\longrightarrow$	M	(FVar)
$\underline{n+1}[M/]$	$\longrightarrow$	$\underline{n}$	(RVar)
$\underline{1}[\Uparrow (S)]$	$\longrightarrow$	$\underline{1}$	(FVarLift)
$\underline{n+1}[\Uparrow (S)]$	$\longrightarrow$	$\underline{n}[S][\uparrow]$	(RVarLift)
$\underline{n}[\uparrow]$	$\longrightarrow$	$\underline{n+1}$	(VarShift)
$(M \odot M')[S]$	$\longrightarrow$	$M[S]\ M'[S]$	(UnMark)
$(M \bullet M')[S]$	$\longrightarrow$	$M[S]\ M'[S]$	(UnMark')

Figure 6: The rewriting system $\lambda^\bullet_\upsilon$

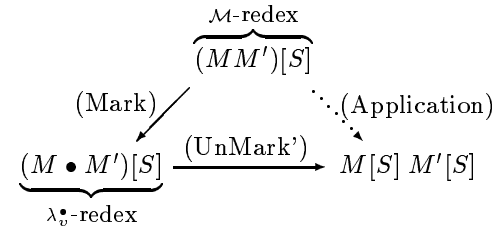
– Case (Mark). There are three cases:

  * Assume that $T = \overbrace{((\lambda M)M')[S]}^{\mathcal{M}\text{-redex}}$, $T' = \overbrace{(\lambda M \bullet M')}^{\lambda^\bullet_\upsilon\text{-redex}}[S]$ and $T'' = M[M'/][S]$ with $T \xrightarrow{\text{(Mark)}} T' \xrightarrow{\text{(Beta}^\bullet\text{)}} T''$. But also $T \xrightarrow{\text{(Beta)}} M[M'/][S]$. Graphically,

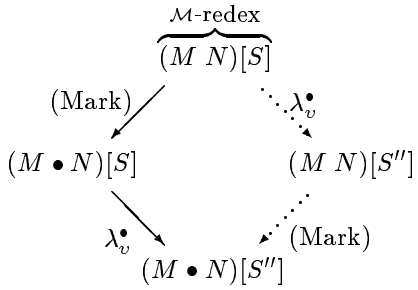
$$\begin{array}{ccc} & \overbrace{((\lambda M)M')[S]}^{\mathcal{M}\text{-redex}} & \\ \text{(Mark)}\swarrow & & \ddots\text{(Beta)} \\ \underbrace{(\lambda M \bullet M')}_{\lambda^\bullet_\upsilon\text{-redex}}[S] & \xrightarrow{\text{(Beta}^\bullet\text{)}} & M[M'/][S] \end{array}$$

  * Assume that $T = \overbrace{(MM')[S]}^{\mathcal{M}\text{-redex}}$, $T' = \overbrace{(M \bullet M')[S]}^{\lambda^\bullet_\upsilon\text{-redex}}$ and $T'' = M[S]\ M'[S]$ with $T \xrightarrow{\text{(Mark)}} T' \xrightarrow{\text{(UnMark')}} T''$. But also $T \xrightarrow{\text{(Application)}} M[S]\ M'[S]$.

Graphically,



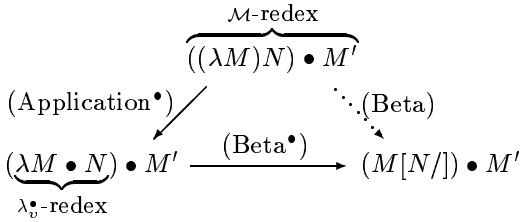
∗ Assume that $T = \overbrace{(M\ N)[S]}^{\mathcal{M}\text{-redex}}$, $T' = (M \bullet N)[S]$ and $T'' = (M \bullet N)[S'']$ with $S \xrightarrow{\lambda_v^\bullet} S''$ and $T \xrightarrow{\text{(Mark)}} T' \xrightarrow{\lambda_v^\bullet} T''$. Also we have that $T \xrightarrow{\lambda_v^\bullet} (M\ N)[S''] \xrightarrow{\text{(Mark)}} (M \bullet N)[S'']$. Graphically,



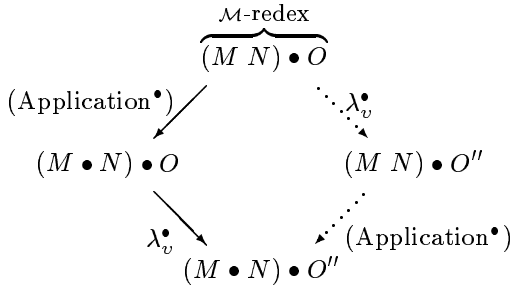
The cases for $M \xrightarrow{\lambda_v^\bullet} M'$ and $N \xrightarrow{\lambda_v^\bullet} N'$ are similar.

– Case (Application•). There are two cases:

∗ Assume that $T = \overbrace{((\lambda M)N) \bullet M'}^{\mathcal{M}\text{-redex}}$, $T' = (\overbrace{\lambda M \bullet N}^{\lambda_v^\bullet\text{-redex}}) \bullet M'$ and $T'' = (M[N/]) \bullet M'$ with $T \xrightarrow{\text{(Application}^\bullet\text{)}} T' \xrightarrow{\text{(Beta}^\bullet\text{)}} T''$. But also $T \xrightarrow{\text{(Beta)}} (M[N/]) \bullet M'$. Graphically,

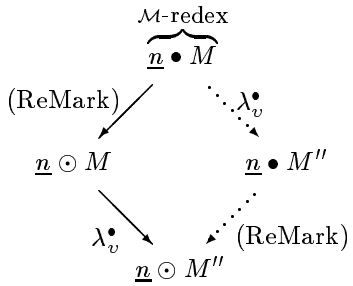


∗ Assume that $T = \overbrace{(M\ N) \bullet O}^{\mathcal{M}\text{-redex}}$, $T' = (M \bullet N) \bullet O$ and $T'' = (M \bullet N) \bullet O''$ with $O \xrightarrow{\lambda_v^\bullet} O''$ and $T \xrightarrow{(\text{Application}^\bullet)} T' \xrightarrow{\lambda_v^\bullet} T''$. Also we have that $T \xrightarrow{\lambda_v^\bullet} (M\ N) \bullet O'' \xrightarrow{(\text{Application}^\bullet)} (M \bullet N) \bullet O''$. Graphically,



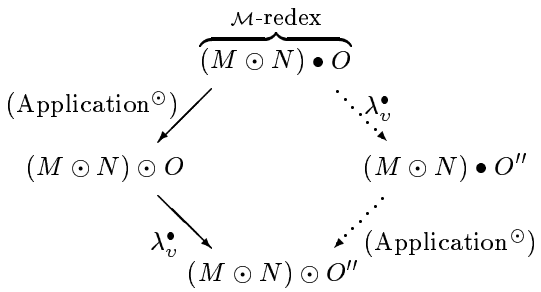
The cases for $M \xrightarrow{\lambda_v^\bullet} M'$ and $N \xrightarrow{\lambda_v^\bullet} N'$ are similar.

– Case (ReMark). Assume that $T = \overbrace{\underline{n} \bullet M}^{\mathcal{M}\text{-redex}}$, $T' = \underline{n} \odot M$ and $T'' = \underline{n} \odot M''$ with $M \xrightarrow{\lambda_v^\bullet} M''$ and $T \xrightarrow{(\text{ReMark})} T' \xrightarrow{\lambda_v^\bullet} T''$. Also we have that $T \xrightarrow{\lambda_v^\bullet} \underline{n} \bullet M'' \xrightarrow{(\text{ReMark})} \underline{n} \odot M''$. Graphically,



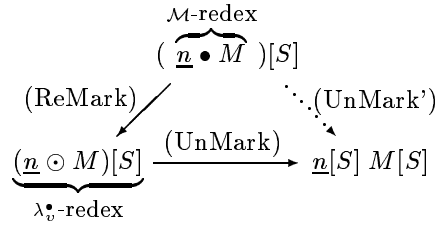
– Case (Application$^\odot$). Assume that $T = \overbrace{(M \odot N) \bullet O}^{\mathcal{M}\text{-redex}}$, $T' = (M \odot N) \odot O$ and $T'' = (M \odot N) \odot O''$ with $O \xrightarrow{\lambda_v^\bullet} O''$ and $T \xrightarrow{(\text{Application}^\odot)} T' \xrightarrow{\lambda_v^\bullet} T''$. Also

we have that $T \xrightarrow{\lambda_v^\bullet} (M \odot N) \bullet O'' \xrightarrow{(\text{Application}^\odot)} (M \odot N) \odot O''$. Graphically,

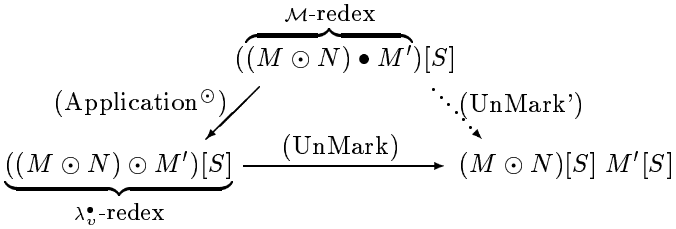


The cases for $M \xrightarrow{\lambda_v^\bullet} M'$ and $N \xrightarrow{\lambda_v^\bullet} N'$ are similar.

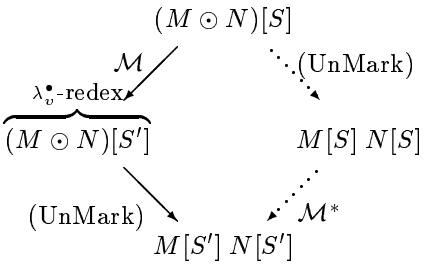
- At the induction step we consider before the cases where T' is a $\lambda_v^\bullet$-redex and then the cases where T is not a $\mathcal{M}$-redex and T' is not a $\lambda_v^\bullet$-redex.
  - Case (UnMark). There are three cases:
    * Assume that $T = (\overbrace{\underline{n} \bullet M}^{\mathcal{M}\text{-redex}})[S]$, $T' = \overbrace{(\underline{n} \odot M)[S]}^{\lambda_v^\bullet\text{-redex}}$ and $T'' = \underline{n}[S]\ M[S]$ with $T \xrightarrow{(\text{ReMark})} T' \xrightarrow{(\text{UnMark})} T''$. But also $T \xrightarrow{(\text{UnMark'})} \underline{n}[S]\ M[S]$. Graphically,



    * Assume that $T = (\overbrace{(M \odot N) \bullet M'}^{\mathcal{M}\text{-redex}})[S]$, $T' = \overbrace{((M \odot N) \odot M')[S]}^{\lambda_v^\bullet\text{-redex}}$ and $T'' = (M \odot N)[S]\ M'[S]$ with $T \xrightarrow{(\text{Application}^\odot)} T' \xrightarrow{(\text{UnMark})} T''$. Also we have that $T \xrightarrow{(\text{UnMark'})} (M \odot N)[S]\ M'[S]$. Graphically,



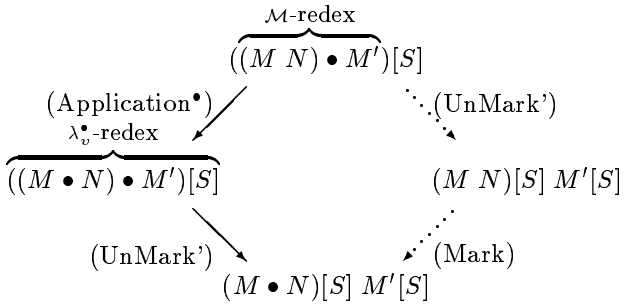
* Assume that $T = (M \odot N)[S]$, $T' = \overbrace{(M \odot N)[S']}^{\lambda_v^\bullet\text{-redex}}$ and $T'' = M[S']\ N[S']$ with $S \xrightarrow{\mathcal{M}} S'$ and $T \xrightarrow{\mathcal{M}} T' \xrightarrow{(\text{UnMark})} T''$. Also we have that $T \xrightarrow{(\text{UnMark})} M[S]\ N[S] \xrightarrow{\mathcal{M}^*} M[S']\ N[S']$. Graphically,



  The cases for $M \xrightarrow{\mathcal{M}} M'$ and $N \xrightarrow{\mathcal{M}} N'$ are similar.

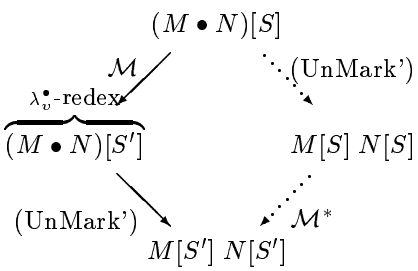
– Case (UnMark'). There are two cases:

* Assume that $T = (\overbrace{(M\ N) \bullet M'}^{\mathcal{M}\text{-redex}})[S]$, $T' = \overbrace{((M \bullet N) \bullet M')[S]}^{\lambda_v^\bullet\text{-redex}}$ and $T'' = (M \bullet N)[S]\ M'[S]$ with $T \xrightarrow{(\text{Application}^\bullet)} T' \xrightarrow{(\text{UnMark'})} T''$. But also $T \xrightarrow{(\text{UnMark'})} (M\ N)[S]\ M'[S] \xrightarrow{(\text{Mark})} (M \bullet N)[S]\ M'[S]$. Graphically,



* Assume that $T = (M \bullet N)[S]$, $T' = \overbrace{(M \bullet N)[S']}^{\lambda_v^\bullet\text{-redex}}$ and $T'' = M[S']\ N[S']$ with $S \xrightarrow{\mathcal{M}} S'$ and $T \xrightarrow{\mathcal{M}} T' \xrightarrow{(\text{UnMark'})} T''$. Also we have that

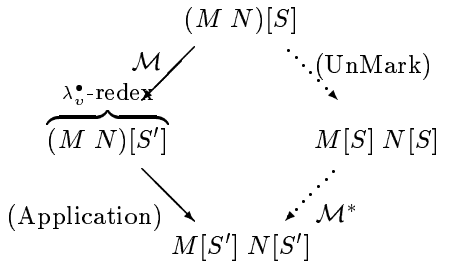
$T \xrightarrow{\text{(UnMark')}} M[S]\ N[S] \xrightarrow{\mathcal{M}^*} M[S']\ N[S']$. Graphically,



The cases for $M \xrightarrow{\mathcal{M}} M'$ and $N \xrightarrow{\mathcal{M}} N'$ are similar.

– Case (Application).

Assume that $T = (M\ N)[S]$, $T' = \overbrace{(M\ N)[S']}^{\lambda_v^\bullet\text{-redex}}$ and $T'' = M[S']\ N[S']$ with $S \xrightarrow{\mathcal{M}} S'$ and $T \xrightarrow{\mathcal{M}} T' \xrightarrow{\text{(Application)}} T''$. Also we have that $T \xrightarrow{\text{(Application)}} M[S]\ N[S] \xrightarrow{\mathcal{M}^*} M[S']\ N[S']$. Graphically,

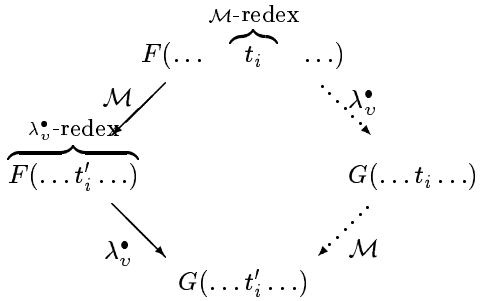


The cases for $M \xrightarrow{\mathcal{M}} M'$ and $N \xrightarrow{\mathcal{M}} N'$ are similar.

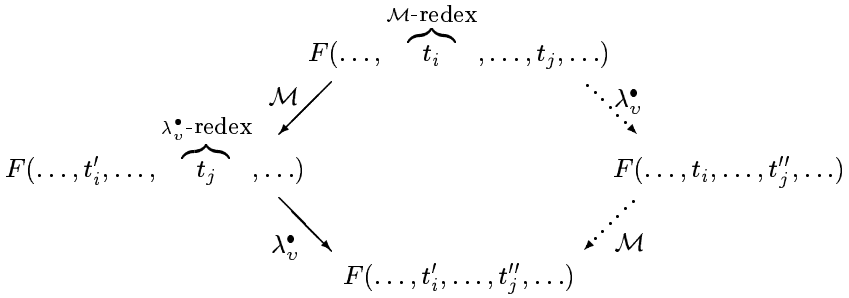
– The rule $\lambda_v^\bullet$ is one of (Beta), (Beta$^\bullet$), (Lambda), (FVar), (RVar), (FVarLift), (RVarLift) and (VarShift). Notice that:

1. All these $\lambda_v^\bullet$-rules are left and right linear.
2. A strict subexpression of one of these $\lambda_v^\bullet$-redexes can never overlaps with a right side of a $\mathcal{M}$-rule.
3. Not any of these $\lambda_v^\bullet$-redexes unifies with a right side a $\mathcal{M}$-rule.

Without lost of generality we assume $T = F(\ldots \overbrace{t_i}^{\mathcal{M}\text{-redex}} \ldots)$, $T' = \overbrace{F(\ldots t_i' \ldots)}^{\lambda_v^\bullet\text{-redex}}$ and $T'' = G(\ldots t_i' \ldots)$ with $t_i \xrightarrow{\mathcal{M}} t_i'$. By the previous remarks the same $\lambda_v^\bullet$-redex of T' exists on T and then we have that $T \xrightarrow{\lambda_v^\bullet} G(\ldots t_i' \ldots)$. But also $G(\ldots t_i \ldots) \xrightarrow{\mathcal{M}} G(\ldots t_i' \ldots)$.

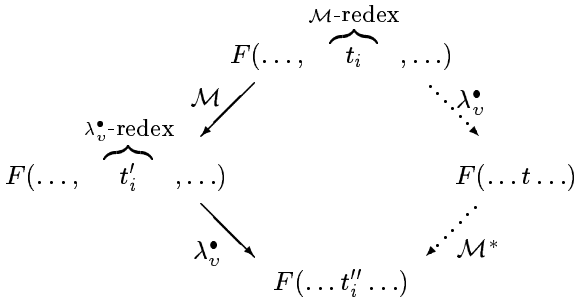


– $\mathcal{M}$ and $\lambda_v^\bullet$ reduce independent sub-terms of T and T'. In this case, we assume $T = F(\ldots, \overbrace{t_i}^{\mathcal{M}\text{-redex}}, \ldots, t_j, \ldots)$, $T' = F(\ldots, t_i', \ldots, \overbrace{t_j}^{\lambda_v^\bullet\text{-redex}}, \ldots)$ and $T'' = F(\ldots, t_i', \ldots, t_j'', \ldots)$ with $t_i \xrightarrow{\mathcal{M}} t_i'$ and $t_j \xrightarrow{\lambda_v^\bullet} t_j''$. But also $T \xrightarrow{\lambda_v^\bullet} F(\ldots, t_i, \ldots, t_j'', \ldots) \xrightarrow{\mathcal{M}} F(\ldots, t_i', \ldots, t_j'', \ldots)$. Graphically,



– $\mathcal{M}$ and $\lambda_v^\bullet$ reduce a common sub-term of T and T'. In this case, we assume $T = F(\ldots, \overbrace{t_i}^{\mathcal{M}\text{-redex}}, \ldots)$, $T' = F(\ldots, \overbrace{t_i'}^{\lambda_v^\bullet\text{-redex}}, \ldots)$ and $T'' = F(\ldots t_i'' \ldots)$ with $t_i \xrightarrow{\mathcal{M}} t_i'$ and $t_i' \xrightarrow{\lambda_v^\bullet} t_i''$. By induction hypothesis there exists t such that $t_i \xrightarrow{\lambda_v^\bullet} t \xrightarrow{\mathcal{M}^*} t_i''$, then we have that $T \xrightarrow{\lambda_v^\bullet} F(\ldots t \ldots) \xrightarrow{\mathcal{M}^*} F(\ldots t_i'' \ldots)$.

Graphically,



□

In order to prove that $\lambda_v^\bullet$ preserves strong normalisation we define an isomorphism which maps $\lambda_v^\bullet$-terms into λ_v-terms:

$$\begin{array}{lcl}
\overline{\underline{n}} & = & \underline{n} \\
\overline{\lambda M} & = & \lambda \overline{M} \\
\overline{M\,M'} & = & \overline{M}\;\overline{M'} \\
\overline{M[S]} & = & \overline{M}[\overline{S}] \\
\overline{M \bullet M'} & = & \overline{M}\;\overline{M'} \\
\overline{M \odot M'} & = & \overline{M}\;\overline{M'} \\
\overline{\uparrow} & = & \uparrow \\
\overline{\Uparrow (S)} & = & \Uparrow (\overline{S}) \\
\overline{M/} & = & \overline{M}/
\end{array}$$

Essentially, this function remove all the marks. Notice that if M is unmarked, then $\overline{M} = M$. It is easy to see that this isomorphism is compatible with the rewriting λ_v-rules in the sense that if $M \xrightarrow{\lambda_v^\bullet} N$ then $\overline{M} \xrightarrow{\lambda_v} \overline{N}$. In fact the new rules of the rewriting system $\lambda_v^\bullet$ with respect to the rewriting system λ_v are (Beta $\bullet$), (UnMark) and (UnMark'), and they correspond exactly to the rules (Beta) the first, and (Application) the last two.

Lemma 28 *If M is strongly λ_v-normalisable then M is strongly $\lambda_v^\bullet$-normalisable.*

Proof. Assume that M is strongly λ_v-normalisable, but it is not strongly $\lambda_v^\bullet$-normalisable. Thus, there is an infinite reduction $M \xrightarrow{\lambda_v^\bullet} M_1 \xrightarrow{\lambda_v^\bullet} \ldots$ and therefore an infinite reduction $\overline{M} \xrightarrow{\lambda_v} \overline{M_1} \xrightarrow{\lambda_v} \ldots$. But also, $\overline{M} = M$ and then there exists an infinite λ_v-reduction of M which is not possible because M is strongly λ_v-normalisable. So we conclude that M is strongly $\lambda_v^\bullet$-normalisable. □

Theorem 4 (Preservation of Strong Normalisation) *λ_ζ preserves strong normalisation.*

Proof. Let M be a ground[6] λ-term such that M is strongly normalisable in λ-calculus. Notice that:

1. λ-terms $\subseteq \lambda_v$-terms $\subseteq \lambda_v^\bullet$-terms $= \lambda_\zeta$-terms , then $M \in \lambda_v$-terms, $M \in \lambda_v^\bullet$-terms and $M \in \lambda_\zeta$-terms.
2. λ_v preserves strong normalisation ([BBLRD95]), then if M is strongly normalisable in λ-calculus, it is strongly λ_v-normalisable.
3. If M is strongly λ_v-normalisable then it is $\lambda_v^\bullet$-strongly normalisable by Lemma 28.
4. If M is strongly $\lambda_v^\bullet$-normalisable then it is $(\mathcal{M} \cup \lambda_v^\bullet)$-strongly normalisable by Lemma 25 using Lemma 26 and Lemma 27 as hypothesis.
5. If M is strongly $(\mathcal{M} \cup \lambda_v^\bullet)$-normalisable then it is strongly λ_ζ-normalisable, since the rewriting system λ_ζ is a subset of $(\mathcal{M} \cup \lambda_v^\bullet)$.

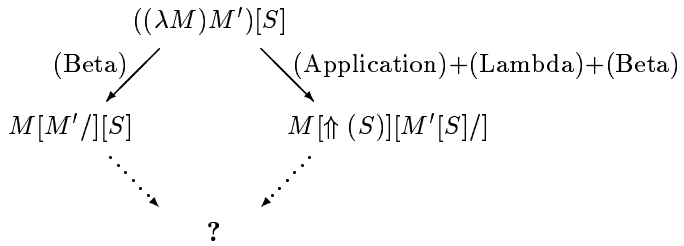
Therefore we conclude that λ_ζ preserves strong normalisation. □

Corollary 4 *Well typed λ-terms are strongly λ_ζ-normalisable.*

7 Conclusions

Calculi of explicit substitutions are very suitable to deal with problems of variable renaming and the introduction of meta-variables. However, they introduce new theoretical questions, for example confluence on open terms. We propose the λ_ζ-system which solves positively the conjecture of the existence of an explicit substitutions calculus confluent on open terms and that preserves strong normalisation.

In order to have a confluent system it is necessary to solve the critical pair generated by the rules (Beta), (Lambda) and (Application):



Thus, we have two choices: either to introduce the composition of substitutions, or to cut one of the branches of this critical pair. Systems like λ_σ and $\lambda_\Uparrow$ implement the first possibility. To implement the latter possibility, we have two natural strategies:

[6]The notion of metavariable does not exist in traditional λ-calculus, then "strong normalisation in open λ-terms" is not a notion well defined

1. cut the right branch, i.e. the reduction of the β-redex before the distribution of the substitution, and

2. cut the left branch, i.e. the distribution of the substitution before the reduction of the β-redex.

The λ_ζ-system can be seen as the coding of the strategy (1) by a first order rewriting system. In order to achieve this, we have modified the rule (Application) to introduce marks in the terms. These marks push substitutions in application terms only if they have a head variable. Thus, for example, we forbid the distribution of a substitutions inside a β-redex. Therefore, we do not have this critical pair and then it is not necessary to introduce the composition operator between substitutions.

Due to this choice, we cannot simulate anymore one step of β-reduction. However, we can prove that our system implements the "big step" semantic of β-reductions, i.e. normal forms of ground λ-terms are the same in both, λ-calculus and λ_ζ-system.

An interesting problem is the implementation of the second strategy, while preserving confluence and strong normalisation in typed terms:

Problem 1 *Does there exist a confluent explicit substitutions calculus that preserves strong normalisation and that reduces substitution redexes before the reduction of a β-redex?*

Simultaneous substitutions, e.g. $M_1 \cdot M_2 \cdots M_n \cdot S$, that have been introduced in some calculi (λ_σ,$\lambda_{\Uparrow}$) by the composition operator, happen to be also useful for other purposes, for example the modelling of the stack of an abstract machine ([HMP95]) and the pruning of search space in unifications algorithms ([DHK95]). Thus we might want to introduce such a feature in λ_ζ. However, re-introduction of composition in λ_ζ, while preserving its properties, does not seem to be easy. For example, if we add the rule $M[S][T] \longrightarrow S \circ T$ we must introduce the general (Application) rule. Thus we state the following problem:

Problem 2 *Does there exist an explicit substitutions calculus confluent (on open terms) that preserves strong normalisation and which accepts reduction under substitutions (not necessarily composition)?*

Acknowledgement

The author is grateful to Gilles Dowek for many helpful discussions and valuable comments on previous versions of this work. We would also like to thank Claude Kirchner, Thérèse Hardin, Paul-André Melliès, Gérard Huet and Jean-Jacques Lévy for their suggestions and encouragement; and last, but not least, Cristina Cifuentes and Cristina Cornes by a careful reading of this paper.

References

[ACCL91] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit substitution. *Journal of Functional Programming*, 1(4):375–416, 1991.

[Bar84] H. P. Barendregt. *The Lambda Calculus, its syntax and semantics*. Studies in Logic and the Foundation of Mathematics. Elsevier Science Publishers B.V. (North-Holland), 1984.

[BBLRD95] Z.-E.-A. Benaissa, D. Briaud, P. Lescanne, and J. Rouyer-Degli. $\lambda\upsilon$, a calculus of explicit substitutions which preserves strong normalisation. Technical Report RR-2477, Unité de recherche INRIA-Lorraine, Janvier 1995.

[CHL95] P.-L. Curien, T. Hardin, and J.-J Lévy. Confluence properties of weak and strong calculi of explicit substitutions. *Journal of the ACM*, 1995. To appear. Also as 1992 INRIA report 1617.

[dB72] N.G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. *Indagationes Mathematicae*, 34(5), 1972.

[DHK95] G. Dowek, T. Hardin, and C. Kirchner. Higher-order unification via explicit substitutions (extended abstract). In *Proceedings, Tenth Annual IEEE Symposium on Logic in Computer Science*, pages 366–374, San Diego, California, 26–29 June 1995. IEEE Computer Society Press.

[Ehr88] T. Ehrhard. *Une sémantique Catégorique des Types Dépendants. Application au Calcul des Constructions*. PhD thesis, U. Paris VII, 1988.

[Her94] H. Herbelin. *Séquents qu'on calcule, de l'interprétation du calcul des séquents comme calcul de λ-termes et comme calcul de stratégies gagnantes*. PhD thesis, U. Paris VII, 1994.

[Her95] H. Herbelin. Notes sur les $\lambda\sigma$-calculs. Personal communication, 1995.

[HMP95] T. Hardin, L. Maranget, and B. Pagano. Functional back-ends and compilers within the lambda-sigma calculus. In Thomas Johnsson, editor, *The Workshop on the Implementation of Functional Languages '95*. Bastad, Sweden, September 1995.

[Hue80] G. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *J.A.C.M.*, 27(4), October 1980.

[KZ89] D. Kapur and H. Zhang. Rrl: A rewrite rule laboratory-user's manual. Technical Report 89-03, Department of Computer Science, The University of Iowa, 1989.

[Les94] P. Lescanne. From $\lambda\sigma$ to $\lambda\upsilon$ a journey through calculi of explicit substitutions. In *Proceedings of the 21st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 60–69, January 1994.

[LRD94] P. Lescanne and J. Rouyer-Degli. The calculus of explicit substitutions $\lambda\upsilon$. Technical report, Centre de Recherche en Informatique de Nancy (CNRS) and INRIA-Lorraine, March 1994.

[Mag94] L. Magnusson. *The Implementation of ALF a Proof Editor based on Martin-Lof's Monomorphic Type Theory with Explicit Substitution.* PhD thesis, Chalmers University of Technology, Department of Computing Science, 1994.

[Mel95] P.-A. Melliès. Typed λ-calculi with explicit substitutions may not terminate. In *Typed Lambda Calculi and Applications*, number 902 in LNCS. Second International Conference TLCA'95, Springer-Verlag, 1995.

[Río93] A. Ríos. *Contributions à l'étude de λ-calculs avec des substitutions explicites.* PhD thesis, U. Paris VII, 1993.

[Saï94] A. Saïbi. Axiomatisation d'un λ-calcul avec substitutions explicites dans coq. Technical Report RR-2345, Unité de recherche INRIA-Rocquencourt, Juillet 1994.

[YH88] H. Yokouchi and T. Hikita. A rewriting system for categorical combinators with multiple arguments. Preprint, 1988.

Unité de recherche Inria Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 Villers Lès Nancy
Unité de recherche Inria Rennes, Irisa, Campus universitaire de Beaulieu, 35042 Rennes Cedex
Unité de recherche Inria Rhône-Alpes, 46 avenue Félix Viallet, 38031 Grenoble Cedex 1
Unité de recherche Inria Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105,
78153 Le Chesnay Cedex
Unité de recherche Inria Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 Sophia-Antipolis Cedex

Éditeur
Inria, Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex (France)
ISSN 0249-6399