A Left-linear Variant of $\lambda\sigma$

César A. Muñoz H.

INRIA Rocquencourt
B.P. 105
78153 Le Chesnay Cedex, France
E-mail: Cesar.Munoz@inria.fr
Tel: (33) 1 39 63 51 57, Fax: (33) 1 39 63 53 30

Abstract. In this paper we consider λ-calculi of explicit substitutions that admit open expressions, i.e. expressions with meta-variables. In particular, we propose a variant of the $\lambda\sigma$-calculus that we call $\lambda_{\mathcal{L}}$. For this calculus and its simply-typed version, we study its meta-theoretical properties. The $\lambda_{\mathcal{L}}$-calculus enjoys the same general characteristics as $\lambda\sigma$, i.e. a simple and finitary first-order presentation, confluent on expressions with meta-variables of terms and weakly normalizing on typed expressions. Moreover, $\lambda_{\mathcal{L}}$ does not have the non-left-linear surjective pairing rule of $\lambda\sigma$ which raises technical problems in some frameworks.

1 Introduction

There are several versions of λ-calculi of explicit substitutions (see, among others, [1, 20, 14, 2, 21, 3, 17, 25, 6]). All these calculi implement β-reductions by means of a lazy mechanism of reduction of substitutions.

In typed λ-calculi, the explicit substitutions have been proposed as a framework for higher-order unification [4, 5, 19], or for representation of incomplete proofs [22, 26]. In these approaches, terms with holes are represented by open terms, i.e. terms with meta-variables.

In order to consider open terms, most of the calculi of explicit substitutions have a strong drawback: non-confluence on terms with meta-variables. Confluence and weak normalization are sufficient to decide equivalence of terms. Hence, these two properties seem to be desirable in any extension of λ-calculi of explicit substitutions with meta-variables.

The $\lambda\sigma$-calculus[1] is one of the most popular calculus of explicit substitutions. It is a first-order rewrite system with two sorts of expressions: terms and substitutions. In this calculus, free and bound variables are represented by de Bruijn's indices, and hence, λ-terms correspond to ground $\lambda\sigma$-terms without substitutions. The $\lambda\sigma$-calculus is not confluent on general open terms [3]. However, it is confluent if we consider expressions with meta-variables of terms but no meta-variables of substitutions [32]. These expressions are usually called *semi-open* expressions.

Compared with other confluent calculi on semi-open expressions (e.g. $\lambda_{\Uparrow}$ [3], $\lambda_{\mathcal{S}_e}$ [14] or λ_{ζ} [25]), the $\lambda\sigma$-calculus is a finitary first-order system ($\lambda_{\mathcal{S}_e}$ is not),

[1] In this paper we use $\lambda\sigma$ to designate the locally confluent calculus proposed in [1].

it allows composition and simultaneous substitutions (λ_ζ does not), and it is compatible with the extensional η-rule ($\lambda_\Uparrow$ is not).

The composition operator was introduced in $\lambda\sigma$ to solve a critical pair, and so, to gain local confluence. Composition of substitutions introduces simultaneous substitutions that happens to be useful for several purposes. For example, the modeling of closures of an abstract machine [12] or the pruning of search space in unification algorithms [4, 5, 19]. Also, this feature improves the substitution mechanism by allowing parallel substitutions of variables. An interesting discussion about composition of substitutions in λ-calculus can be found in [29].

However, composition of substitutions and simultaneous substitutions are responsible of the following non-left-linear rule in $\lambda\sigma$: $\mathbf{1}[S] \cdot (\uparrow \circ S) \xrightarrow{\text{(SCons)}} S$. Informally, if we interpret S as a list, $\mathbf{1}$ as the head function and $\uparrow$ as the tail function, then this rule corresponds to the surjective-pairing rule. The (SCons)-rule is impractical for many reasons. We have shown in [27] that $\lambda\sigma$ may loses the subject reduction property in a dependent type system due to (SCons). But also, independently, Nadathur [30] has remarked that this non-left-linear rule is difficult to handle in implementations. In fact, he shows that (SCons) is admissible in $\lambda\sigma$ when we consider semi-open terms and the following scheme of rule: $\mathbf{1}[\uparrow^n] \cdot \uparrow^{n+1} \xrightarrow{\text{(SCons)}} \uparrow^n$, where $\uparrow^n$ is a notation for $\overbrace{\uparrow \circ \ldots \circ \uparrow}^{n\text{-times}}$.

Following this idea, we propose a calculus of explicit substitutions that enjoys the same general features as $\lambda\sigma$, i.e. a simple and finitary first-order presentation, confluent on expressions with meta-variables of terms and weakly normalizing on typed terms. But, in contrast to $\lambda\sigma$, the new calculus does not have the (SCons)-rule which raises technical problems in some frameworks.

The rest of the paper is organized as follows. In Section 2 we present the $\lambda_{\mathcal{L}}$-calculus. The confluence property of $\lambda_{\mathcal{L}}$ is show in Section 3. In Section 4 we study the simply-typed version of $\lambda_{\mathcal{L}}$. In Section 5 we prove that $\lambda_{\mathcal{L}}$ is weakly normalizing on typed expressions. Last section summarizes the main contributions of this work.

2 $\lambda_{\mathcal{L}}$-Calculus

The finitary presentation of the scheme suggested by Nadathur is gained by the introduction of a sort to represent natural numbers and with an adequate set of rewrite rules to compute with them.

Well formed expressions in $\lambda_{\mathcal{L}}$ are defined by the following grammar:[2]

$$\begin{array}{lll}
\textbf{Naturals} & n & ::= \ 0 \mid Suc(n) \\
\textbf{Terms} & M, N & ::= \ \mathbf{1} \mid \lambda M \mid (M\ N) \mid M[S] \\
\textbf{Substitutions} & S, T & ::= \ \uparrow^n \ \mid M \cdot S \mid S \circ T
\end{array}$$

[2] In previous manuscripts ([28, 27]) the name of the calculus was λ_ϕ, but we have changed to $\lambda_{\mathcal{L}}$ in order to avoid confusion with the $\lambda\phi$-calculus proposed by Lescanne in [20].

The $\lambda_{\mathcal{L}}$-calculus is given by the rewrite system in Fig. 1.

$(\lambda M N)$	$\longrightarrow$	$M[N \cdot \uparrow^0]$	(Beta)
$(\lambda M)[S]$	$\longrightarrow$	$\lambda M[\mathbf{1} \cdot (S \circ \uparrow^{Suc(0)})]$	(Lambda)
$(M\ N)[S]$	$\longrightarrow$	$M[S]\ N[S]$	(App)
$M[S][T]$	$\longrightarrow$	$M[S \circ T]$	(Clos)
$\mathbf{1}[M \cdot S]$	$\longrightarrow$	M	(VarCons)
$M[\uparrow^0]$	$\longrightarrow$	M	(Id)
$(M \cdot S) \circ T$	$\longrightarrow$	$M[T] \cdot (S \circ T)$	(Map)
$\uparrow^0 \circ S$	$\longrightarrow$	S	(IdS)
$\uparrow^{Suc(n)} \circ (M \cdot S)$	$\longrightarrow$	$\uparrow^n \circ S$	(ShiftCons)
$\uparrow^{Suc(n)} \circ \uparrow^m$	$\longrightarrow$	$\uparrow^n \circ \uparrow^{Suc(m)}$	(ShiftShift)
$\mathbf{1} \cdot \uparrow^{Suc(0)}$	$\longrightarrow$	$\uparrow^0$	(Shift0)
$\mathbf{1}[\uparrow^n] \cdot \uparrow^{Suc(n)}$	$\longrightarrow$	$\uparrow^n$	(ShiftS)

Fig. 1. The rewrite system $\lambda_{\mathcal{L}}$

A first remark is that $\lambda_{\mathcal{L}}$ gathers, in its syntax, some notations that are frequently used to speak informally of $\lambda\sigma$. For example, the substitutions *id* and $\overbrace{\uparrow \circ \ldots \circ \uparrow}^{n+1}$ of $\lambda\sigma$, correspond respectively to $\uparrow^0$ and $\uparrow^{Suc(n)}$ in $\lambda_{\mathcal{L}}$. In the same way, the de Bruijn's indices $\underline{1}$ and $\underline{n+1}$ are represented respectively by $\mathbf{1}$ and $\mathbf{1}[\uparrow^n]$ in $\lambda_{\mathcal{L}}$. Thus, the scheme of rule suggested by Nadathur can be written as a first-order (and finitary) rule.

The sub-system $\mathcal{L}$ is obtained by dropping (Beta) from $\lambda_{\mathcal{L}}$.

Proposition 1. *$\mathcal{L}$ is terminating.*

Proof. In [36] this property has been proved by using the *semantic labelling* method (cf. [35]). An alternative proof is proposed in [28]. □

The rewrite system $\mathcal{L}$ is not confluent, not even locally confluent, on open expressions. We can check mechanically, for example using the RRL system [16], that $\mathcal{L}$ has the following critical pairs:

- **(Id-Clos)**. $M[S] \xleftarrow{\mathcal{L}^+} M[S][\uparrow^0] \xrightarrow{\mathcal{L}^+} M[S \circ \uparrow^0]$.
- **(Clos-Clos)**. $M[(S \circ T) \circ T'] \xleftarrow{\mathcal{L}^+} M[S][T][T'] \xrightarrow{\mathcal{L}^+} M[S \circ (T \circ T')]$.
- **(Shift0-Map)**. $S \xleftarrow{\mathcal{L}^+} (\mathbf{1} \cdot \uparrow^{Suc(0)}) \circ S \xrightarrow{\mathcal{L}} \mathbf{1}[S] \cdot (\uparrow^{Suc(0)} \circ S)$.
- **(ShiftS-Map)**. $\uparrow^n \circ S \xleftarrow{\mathcal{L}} (\mathbf{1}[\uparrow^n] \cdot \uparrow^{Suc(n)}) \circ S \xrightarrow{\mathcal{L}^+} \mathbf{1}[\uparrow^n \circ S] \cdot (\uparrow^{Suc(n)} \circ S)$.

– **(Lambda-Clos).** $\lambda M[\mathbf{1} \cdot ((S \circ \uparrow^{Suc(0)}) \circ (\mathbf{1} \cdot (T \circ \uparrow^{Suc(0)})))] \xleftarrow{\mathcal{L}^+}$
$(\lambda M)[S][T] \xrightarrow{\mathcal{L}^+} \lambda M[\mathbf{1} \cdot ((S \circ T) \circ \uparrow^{Suc(0)})]$.

If we consider (Beta), then we have additionally the following critical pair with (App):

$$M[N[S] \cdot S] \xleftarrow{\lambda_{\mathcal{L}}{}^+} (\lambda M\ N)[S] \xrightarrow{\lambda_{\mathcal{L}}{}^+} M[N[S] \cdot ((S \circ \uparrow^{Suc(0)}) \circ (N \cdot \uparrow^0))]$$

The following lemma proves that these critical pairs are joinable on the set of expressions that contain meta-variables of terms, but no meta-variables of substitutions or naturals, i.e. on semi-open expressions. Due to space limitation, we only sketch the proof of the main properties. For detailed proofs, see the extended version of this paper [28].

Proposition 2. *The critical pairs of $\lambda_{\mathcal{L}}$ are $\mathcal{L}$-joinable on semi-open expressions.*

Proof. For any any critical pair we reduce substitutions to $\mathcal{L}$-normal forms. Next, we proceed by structural induction on $\mathcal{L}$-normal substitutions. □

In the general case, local ground confluence cannot be mechanically verified [15]. However, the Critical Pair's lemma [13], i.e. a rewrite system is confluent if its critical pairs are joinable, holds also for ground expressions (cf. [15]): a rewrite system is *ground confluent* if its critical pairs are *ground joinable*. The Critical Pair's Lemma is not true for general many-sorted systems, but in [33] it has been proved that it holds if for every rule $l \longrightarrow r$, the sort of l and the sort of r are the same.

Proposition 3. *$\mathcal{L}$ is locally confluent on semi-open expressions.*

Proof. Notice that the $\lambda_{\mathcal{L}}$-calculus has three sorts of expressions: Naturals, Substitutions and Terms, but only meta-variables of terms are admitted. We must extend the Critical Pair's lemma to semi-open expressions. We check that $\mathcal{L}$ is a sort compatible system, i.e. terms reduce to terms and substitutions reduce to substitutions. Now, the proof follows straightforwardly the proofs in [33, 15]. Notice that if two expressions are joinable, then they are in particular joinable in semi-open expressions. Hence, it suffices to concentrate on those critical pairs that are not joinable on open terms, and we conclude with Proposition 2. □

Theorem 4. *$\mathcal{L}$ is confluent on semi-open expressions.*

Proof. By Proposition 1, $\mathcal{L}$ is terminating and by Proposition 3, $\mathcal{L}$ is locally confluent, so by Newman's Lemma, $\mathcal{L}$ is confluent. □

Corollary 5. *$\mathcal{L}$-normal forms of semi-open expressions always exist, and they are unique. We denote by $x{\downarrow}_{\mathcal{L}}$ the $\mathcal{L}$-normal form of x.*

Remark: The non-linearity of $\lambda_{\mathcal{L}}$ due to (ShiftS) is only apparent since the term with a double occurrence in this rule can be considered as a constant in the set of semi-open expressions. In particular, there are not reduction rules for natural numbers.

3 Confluence

An useful technique to prove confluence in calculi of explicit substitutions is the *interpretation method* [11, 17]. Although the interpretation method can be used to prove confluence on terms with meta-variables (cf. [32]), we prefer to use a technique that was coined in [34]: the Yokouchi-Hikita's Lemma. This lemma seems to be suitable for left-linear calculi of explicit substitutions [3, 31, 25].

Lemma 6 Yokouchi-Hikita's Lemma. *Let R and S be two relations defined on a set X such that: 1. R is confluent and terminating, 2. S is strongly confluent and 3. S and R commute in the following way, for any $x, y, z \in X$, if $x \xrightarrow{R} y$ and $x \xrightarrow{S} z$, then there exists $w \in X$ such that $y \xrightarrow{R^*SR^*} w$ and $z \xrightarrow{R^*} w$, i.e. the following diagram holds:*

$$\begin{tikzcd}
 & x \arrow[dl, "R"'] \arrow[dr, "S"] & \\
y \arrow[dr, dotted, "R^*SR^*"'] & & z \arrow[dl, dotted, "R^*"] \\
 & w &
\end{tikzcd}$$

Then the relation R^*SR^* is confluent.

Proof. See [3]. □

We take the set of semi-open expressions as X, $\mathcal{L}$ as R and $B_{\|}$ as S, where $B_{\|}$ is the parallelization of (Beta) defined by:

$$\frac{}{x \longrightarrow x}\,(\mathrm{Refl}_{\|}) \qquad \frac{M \longrightarrow N}{\lambda M \longrightarrow \lambda N}\,(\mathrm{Lambda}_{\|})$$

$$\frac{M \longrightarrow M' \qquad N \longrightarrow N'}{M\ N \longrightarrow M'\ N'}\,(\mathrm{App}_{\|}) \qquad \frac{M \longrightarrow N \qquad S \longrightarrow T}{M[S] \longrightarrow N[T]}\,(\mathrm{Clos}_{\|})$$

$$\frac{M \longrightarrow N \qquad S \longrightarrow T}{M \cdot S \longrightarrow N \cdot T}\,(\mathrm{Cons}_{\|}) \qquad \frac{S \longrightarrow S' \qquad T \longrightarrow T'}{S \circ T \longrightarrow S' \circ T'}\,(\mathrm{Comp}_{\|})$$

$$\frac{M \longrightarrow M' \qquad N \longrightarrow N'}{(\lambda M\ N) \longrightarrow M'[N' \cdot \uparrow^0]}\,(\mathrm{Beta}_{\|})$$

Proposition 7. *On semi-open expressions, $\mathcal{L}$ and $B_{\|}$ satisfy the conditions of Lemma 6. Therefore, $\mathcal{L}^* B_{\|} \mathcal{L}^*$ is confluent.*

Proof. (1) By Proposition 1 and Theorem 4, $\mathcal{L}$ is terminating and confluent on semi-open expressions. (2) $B_{\|}$ is strongly confluent, since (Beta) by itself is a left linear system with no critical pairs (cf. [13]). (3) Assume that an arbitrary expression x reduces in one $\mathcal{L}$-step to y, and in one $B_{\|}$-step to z. We prove, by induction on the depth of the $\mathcal{L}$-redex reduced in x, that there exists w such that $y \xrightarrow{\mathcal{L}^* B_{\|} \mathcal{L}^*} w$ and $z \xrightarrow{\mathcal{L}^*} w$. At the base case x is a $\mathcal{L}$-redex:

- (App). There are two cases:
  - $x = (M\ N)[S] \xrightarrow{\text{(App)}} (M[S]\ N[S]) = y$ and $(M\ N)[S] \xrightarrow{B_{\|}} (M'\ N')[S'] = z$, with $M \xrightarrow{B_{\|}} M'$, $N \xrightarrow{B_{\|}} N'$ and $S \xrightarrow{B_{\|}} S'$. By definition of $B_{\|}$, $(M[S]\ \ N[S]) \xrightarrow{B_{\|}} (M'[S']\ \ N'[S']) = w$. But also, $(M'\ N')[S'] \xrightarrow{\text{(App)}} (M'[S']\ N'[S']) = w$.
  - $x = (\lambda M\ N)[S] \xrightarrow{\text{(App)}} ((\lambda M)[S]\ N[S]) = y$ and $(\lambda M\ N)[S] \xrightarrow{B_{\|}} M'[N' \cdot \uparrow^0][S'] = z$, with $M \xrightarrow{B_{\|}} M'$, $N \xrightarrow{B_{\|}} N'$ and $S \xrightarrow{B_{\|}} S'$. Let $\hat{S}'$ the $\mathcal{L}$-normal form of S' (Corollary 5). Then, $y = ((\lambda M)[S]\ N[S]) \xrightarrow{\text{(Lambda)}} (\lambda M[\mathbf{1} \cdot (S \circ \uparrow^{Suc(0)})]\ N[S]) \xrightarrow{B_{\|}}$ $M'[\mathbf{1} \cdot (S' \circ \uparrow^{Suc(0)})][N'[S'] \cdot \uparrow^0] \xrightarrow{\mathcal{L}^*} M'[N'[\hat{S}'] \cdot \hat{S}']$. But also, $M'[N' \cdot \uparrow^0][S'] \xrightarrow{\mathcal{L}^*} M'[N'[\hat{S}'] \cdot \hat{S}']$. This case is the only interesting one.
- (Lambda). $x = (\lambda M)[S] \xrightarrow{\text{(Lambda)}} \lambda M[\mathbf{1} \cdot (S \circ \uparrow^{Suc(0)})] = y$ and $x = (\lambda M)[S] \xrightarrow{B_{\|}} (\lambda M')[S'] = z$, with $M \xrightarrow{B_{\|}} M'$ and $S \xrightarrow{B_{\|}} S'$. By definition of $B_{\|}$, $\lambda M[\mathbf{1} \cdot (S \circ \uparrow^{Suc(0)})] \xrightarrow{B_{\|}} \lambda M'[\mathbf{1} \cdot (S' \circ \uparrow^{Suc(0)})] = w$. But also, $(\lambda M')[S'] \xrightarrow{\text{(Lambda)}} \lambda M'[\mathbf{1} \cdot (S' \circ \uparrow^{Suc(0)})] = w$.
- The other cases are similar to the previous one.

At the induction step we solve with the induction hypothesis. □

Theorem 8 Confluence. *$\lambda_{\mathcal{L}}$ is confluent on semi-open expressions.*

Proof. Notice that $\lambda_{\mathcal{L}} \subseteq \mathcal{L}^* B_{\|} \mathcal{L}^* \subseteq {\lambda_{\mathcal{L}}}^*$. If $x \xrightarrow{{\lambda_{\mathcal{L}}}^*} y$ and $x \xrightarrow{{\lambda_{\mathcal{L}}}^*} z$, then by Proposition 7, there exists w such that $y \xrightarrow{(\mathcal{L}^* B_{\|} \mathcal{L}^*)^*} w$ and $z \xrightarrow{(\mathcal{L}^* B_{\|} \mathcal{L}^*)^*} w$. So, $y \xrightarrow{{\lambda_{\mathcal{L}}}^*} w$ and $z \xrightarrow{{\lambda_{\mathcal{L}}}^*} w$. □

4 The simply-typed version

We consider a simple type theory, where types are generated from a set of basic types $a, b, \ldots$ and the arrow $(\rightarrow)$ type constructor. The simple type system we propose is inspired in that of $\lambda\sigma$ [1].

Like the simply-typed λ-calculus in de Bruijn's notation, typing contexts (of free variables) are structured as lists of types. The grammar of types and contexts is:

$$\begin{array}{lll} \textbf{Types}\ A, B & ::= & a, b, \ldots \mid A \rightarrow B \\ \textbf{Contexts}\ \Gamma & ::= & nil \mid A.\Gamma \end{array}$$

Typed terms differ from untyped ones only in abstraction expressions. We prefer a *Church style* notation where types of binder variables appear explicitly in the syntax.

$$\textbf{Terms}\ M, N \ ::= \ \ldots \mid \lambda_A.M \mid \ldots$$

The $\lambda_{\mathcal{L}}$-calculus is modified according to this new syntax of abstractions. However, it is not difficult to see that properties of Section 2 and 3 are preserved.

Typing assertions have one of the following forms:

- $\Gamma \vdash M : A$, the term M has type A in the context Γ.
- $\Gamma \vdash S \triangleright \Delta$, the substitution S has type Δ in the context Γ.

$$\frac{}{A.\Gamma \vdash \mathbf{1} : A}\ (\text{Var}) \qquad \frac{A.\Gamma \vdash M : B}{\Gamma \vdash \lambda_A.M : A \to B}\ (\text{Abs})$$

$$\frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash (M\ N) : B}\ (\text{Appl}) \qquad \frac{\Gamma \vdash S \triangleright \Delta \quad \Delta \vdash M : A}{\Gamma \vdash M[S] : A}\ (\text{Clos})$$

$$\frac{}{\Gamma \vdash \uparrow^0 \triangleright \Gamma}\ (\text{Id}) \qquad \frac{\Gamma \vdash \uparrow^n \triangleright \Delta}{A.\Gamma \vdash \uparrow^{Suc(n)} \triangleright \Delta}\ (\text{Shift})$$

$$\frac{\Gamma \vdash S \triangleright \Delta' \quad \Delta' \vdash T \triangleright \Delta}{\Gamma \vdash T \circ S \triangleright \Delta}\ (\text{Comp}) \qquad \frac{\Gamma \vdash M : A \quad \Gamma \vdash S \triangleright \Delta}{\Gamma \vdash M \cdot S \triangleright A.\Delta}\ (\text{Cons})$$

Each meta-variable is typed in a unique context by a unique type (c.f. [4, 22]):

$$\frac{}{\Gamma_X \vdash X : A_X}\ (\text{Meta}_X)$$

Example 1.

1. This is a type derivation of $A.nil \vdash \lambda_B.(X\ \mathbf{1}[\uparrow^{Suc(0)}]) : B \to C$.

$$\frac{\dfrac{\dfrac{}{B.A.nil \vdash X : A \to C}\ (\text{Meta}_X) \quad \dfrac{\dfrac{\dfrac{}{A.nil \vdash \uparrow^0 \triangleright A.nil}\ (\text{Id})}{B.A.nil \vdash \uparrow^{Suc(0)} \triangleright A.nil}\ (\text{Shift}) \quad \dfrac{}{A.nil \vdash \mathbf{1} : A}\ (\text{Var})}{B.A.nil \vdash \mathbf{1}[\uparrow^{Suc(0)}] : A}\ (\text{Clos})}{B.A.nil \vdash (X\ \mathbf{1}[\uparrow^{Suc(0)}]) : C}\ (\text{Appl})}{A.nil \vdash \lambda_B.(X\ \mathbf{1}[\uparrow^{Suc(0)}]) : B \to C}\ (\text{Abs})$$

2. The term $(\lambda_A.X\ X)$ is not well-typed in any context. Notice that in the following derivation:

$$\frac{\dfrac{\dfrac{\ldots}{A.\Gamma \vdash X : A}}{\Gamma \vdash \lambda_A.X : A \to A}\ (\text{Abs}) \qquad \dfrac{\ldots}{\Gamma \vdash X : A}}{\Gamma \vdash (\lambda_A.X\ X) : A \to A}\ (\text{Appl})$$

the meta-variable X must be typed in two different contexts: $A.\Gamma$ and Γ.

3. Let X be a meta-variable such that $\Gamma \vdash X : A$. In this example, we take the index $\underline{2}$ as a notation for $\mathbf{1}[\uparrow^{Suc(0)}]$. We have the valid typing judgment: $\Gamma \vdash (\lambda_A.\lambda_B.\underline{2}\, X) : B \to A$. We obtain by $\lambda_{\mathcal{L}}$-reduction:
$$(\lambda_A.\lambda_B.\underline{2}\, X) \xrightarrow{\text{(Beta)}} (\lambda_B.\underline{2})[X \cdot \uparrow^0] \xrightarrow{\lambda_{\mathcal{L}}{}^*} \lambda_B.X[\uparrow^{Suc(0)}]$$
Also, we can verify that $\Gamma \vdash \lambda_B.X[\uparrow^{Suc(0)}] : B \to A$.

Notice that the type system is syntax directed, i.e. there is one rule for each constructor of terms and substitutions. Using this fact, we can prove easily that for a given context, the type of an expression is unique (*type uniqueness' lemma*).

Lemma 9 Type Uniqueness.

1. If $\Gamma_1 \vdash M : A_1$ and $\Gamma_2 \vdash M : A_2$, then $A_1 = A_2$.
2. If $\Gamma_1 \vdash S \triangleright \Delta_1$ and $\Gamma_2 \vdash S \triangleright \Delta_2$, then $\Delta_1 = \Delta_2$.

Proof. We proceed by simultaneous structural induction on M and S. □

Example 1(3) suggests that typing is preserved under $\lambda_{\mathcal{L}}$-reductions. This property is known as *subject reduction*.

Theorem 10 Subject Reduction. *Let x and y be such that $x \xrightarrow{\lambda_{\mathcal{L}}{}^*} y$, then*

– *if x is a term and $\Gamma \vdash x : A$, then $\Gamma \vdash y : A$, and*
– *if x is a substitution and $\Gamma \vdash x \triangleright \Delta$, then $\Gamma \vdash y \triangleright \Delta$.*

Proof. We show that typing is preserved for one-step reductions (i.e. $\xrightarrow{\lambda_{\mathcal{L}}}$), and then it is also for its reflexive and transitive closure (i.e. $\xrightarrow{\lambda_{\mathcal{L}}{}^*}$). Let $x \xrightarrow{\lambda_{\mathcal{L}}} y$ be a one-step reduction, we proceed by induction on the depth of the redex reduced in x. At the initial case x is reduced at the top level, and we prove that every rule preserves typing. At the induction step we resolve with induction hypothesis. □

In the $\lambda_{\mathcal{L}}$-system, just as in $\lambda\sigma$, instantiation of meta-variables and typing commute. This property guarantees the soundness of instantiation of meta-variables in the unification algorithm [4, 5, 19], or in the refinements steps of incomplete proofs [26].

Lemma 11 Instantiation Soundness. *Let N be a term such that $\Gamma_X \vdash N : A_X$, where Γ_X and A_x are respectively the unique context and unique type of a meta-variable X. Then,*

1. if $\Delta \vdash M : B$, then $\Delta \vdash M\{X \mapsto N\} : B$, and
2. if $\Delta \vdash S \triangleright \Delta'$, then $\Delta \vdash S\{X \mapsto N\} \triangleright \Delta'$,

where $x\{X \mapsto N\}$ is a notation for the remplacement of meta-variable X by N in the expression x without take care of possible capture of free variables.

Proof. We reason by induction on type derivation. □

5 Weak Normalization

Strong normalization on typed terms does not hold for $\lambda_{\mathcal{L}}$. In fact, Melliès shows in [23] that his counter-example for preservation of strong normalization in the $\lambda\sigma$-calculus [24], can be adapted to systems without associativity of composition (as $\lambda_{\mathcal{L}}$), and even if we give priority to the rules (ShiftCons) and (VarCons).

In λ-calculi of explicit substitutions that implement one-step semantic of β-reduction —i.e. if M, N are pure terms[3] and $M \xrightarrow{\beta} N$, then $M \xrightarrow{\text{(Beta)}} M'$ where N is the substitution-normal form of M'— as $\lambda\sigma$, $\lambda_{\Uparrow}$ and $\lambda_{\mathcal{L}}$, weak normalization on typed pure terms follows directly from strong normalization of typed λ-calculus. When we consider semi-open expressions, it arises an additional difficulty: the presence of meta-variables and substitutions on normal forms. Notice that the set of normal forms of semi-open expressions is not include in the set of pure terms, e.g. the term $X[\uparrow^{Suc(0)}]$ is a $\lambda_{\mathcal{L}}$-normal form, but it is not pure.

For the simply-typed version of $\lambda\sigma$ (with meta-variables), Goubault-Larreq [10] proposes a clever translation from $\lambda\sigma$-terms into a family of λ-terms. In this approach, weak normalization is deduced from strong normalization of the simply-typed λ-calculus. That proof is adapted to a second-order type system without dependent types in [9].

In this section, we prove that $\lambda_{\mathcal{L}}$ is weakly normalizing on typed expressions. In particular, we show that the reduction of (Beta) followed by a $\mathcal{L}$-normalization is strongly normalizing on typed expressions. The proof we provide can be adapted to $\lambda\sigma$ in a straightforward way. This gives an alternative proof to that developed by Goubault-Larreq. Our proof is based on that proposed by Geuvers for the Calculus of Construction [7]. The technique that we use is extended to a dependent type system with explicit substitutions in [27].

The general idea of the proof is to give an interpretation for each type into a set of terms satisfying certain closure properties (these sets are called *saturated* sets). Terms are also interpreted by functions called *valuations*. In our proof, valuations are just particular explicit substitutions. We prove that if M is a $\mathcal{L}$-normal form and $\Gamma \vdash M : A$, then for any valuation S of M, the substitution normal form of $M[S]$, i.e. $(M[S])\downarrow_{\mathcal{L}}$, is included in the interpretation of A, denoted $[\![A]\!]$. The identity substitution is a valuation of any term, thus, in particular, $(M[\uparrow^0])\downarrow_{\mathcal{L}} = M \in [\![A]\!]$. The closure properties of $[\![A]\!]$ are sufficient to conclude that M is weakly normalizing.

We define $\mathcal{NF}_{\mathcal{L}}$ as the set that contains all the $\mathcal{L}$-normal forms of semi-open expressions.

Definition 12. Let $x, y \in \mathcal{NF}_{\mathcal{L}}$, we say that x *$\beta_{\mathcal{L}}$-converts* to y, noted by $x \xrightarrow{\beta_{\mathcal{L}}} y$, if and only if $x \xrightarrow{\text{(Beta)}} w$ and $y = w\downarrow_{\mathcal{L}}$.

We denote by $\mathcal{SN}$ the set of $\beta_{\mathcal{L}}$-strongly normalizing expressions of $\mathcal{NF}_{\mathcal{L}}$.

Definition 13. Let M be in $\mathcal{NF}_{\mathcal{L}}$, M is *neutral* if it does not have the form $\lambda_A.N$. The set of neutral terms is denoted by $\mathcal{NT}$.

[3] A pure term is a ground term which does not contain substitutions.

Definition 14. A set of terms $\Lambda \subseteq \mathcal{NF}_{\mathcal{L}}$ is *saturated* if

1. $\Lambda \subseteq \mathcal{SN}$.
2. If $M \in \Lambda$ and $M \xrightarrow{\beta_{\mathcal{L}}} M'$, then $M' \in \Lambda$.
3. If $M \in \mathcal{NT}$, and whenever we reduce a $\beta_{\mathcal{L}}$-redex of M we obtain a term $M' \in \Lambda$, then $M \in \Lambda$.

The set of saturated sets is denoted by **SAT**.

From Def. 14(3):

Remark 15. Let $M \in \mathcal{NT}$ such that M is a $\beta_{\mathcal{L}}$-normal form. For any $\Lambda \in \mathbf{SAT}$, $M \in \Lambda$.

Lemma 16. $\mathcal{SN} \in \mathbf{SAT}$.

Proof. We verify easily the following conditions.

1. $\mathcal{SN} \subseteq \mathcal{SN}$.
2. If $M \in \mathcal{SN}$ and $M \xrightarrow{\beta_{\mathcal{L}}} M'$, then $M' \in \mathcal{SN}$.
3. If $M \in \mathcal{NT}$, and whenever we reduce a $\beta_{\mathcal{L}}$-redex of M we obtain a term $M' \in \mathcal{SN}$, then $M \in \mathcal{SN}$.

□

Definition 17. Let $\Lambda, \Lambda' \in \mathbf{SAT}$, we define the set

$$\Lambda \to \Lambda' = \{M \in \mathcal{NF}_{\mathcal{L}} \mid \forall N \in \Lambda \; : \; (M\ N) \in \Lambda'\}$$

Lemma 18. **SAT** *is closed under function spaces, i.e. if* $\Lambda, \Lambda' \in \mathbf{SAT}$*, then* $\Lambda \to \Lambda' \in \mathbf{SAT}$.

Proof. We show:

1. $\Lambda \to \Lambda' \subseteq \mathcal{SN}$.
   Let $M \in \Lambda \to \Lambda'$, by Def. 17 and Def. 14(1), $(M\ N) \in \Lambda' \subseteq \mathcal{SN}$ for all $N \in \Lambda$. Thus, $M \in \mathcal{SN}$.
2. If $M \in \Lambda \to \Lambda'$ and $M \xrightarrow{\beta_{\mathcal{L}}} M'$, then $M' \in \Lambda \to \Lambda'$.
   Let $N \in \Lambda$, we show that $(M'\ N) \in \Lambda'$. By hypothesis, $(M\ N) \xrightarrow{\beta_{\mathcal{L}}} (M'\ N)$, and $(M\ N) \in \Lambda'$. Thus, by Def. 14(2), $(M'\ N) \in \Lambda'$.
3. If $M \in \mathcal{NT}$, and whenever we reduce a $\beta_{\mathcal{L}}$-redex of M we obtain a term $M' \in \Lambda \to \Lambda'$, then $M \in \Lambda \to \Lambda'$.
   Let $N \in \Lambda$, we show that $(M\ N) \in \Lambda'$. Since $(M\ N) \in \mathcal{NT}$, then by Def. 14(3), it suffices to prove that if $(M\ N) \xrightarrow{\beta_{\mathcal{L}}} M''$, then $M'' \in \Lambda'$. We have $N \in \Lambda \subseteq \mathcal{SN}$, so we can reason by induction on $\nu(N)$[4]. In one step $(M\ N)$ $\beta_{\mathcal{L}}$-reduces to:

[4] "If x is strongly normalizing, $\nu(x)$ is a number which bounds the length of every normalization sequence beginning with x" [8].

- $(M'\ N)$, with $M \xrightarrow{\beta_{\mathcal{L}}} M'$. By hypothesis, $M' \in \Lambda \to \Lambda'$ and $N \in \Lambda$, thus $(M'\ N) \in \Lambda'$.
- $(M\ N')$, with $N \xrightarrow{\beta_{\mathcal{L}}} N'$. By Def. 14(2), $N' \in \Lambda$, and $\nu(N') < \nu(N)$, so by induction hypothesis, $(M\ N') \in \Lambda'$.
- There is no other possibility since $M \in \mathcal{NT}$.

□

Definition 19. The *type interpretation function* is defined inductively on types as follows:

$$\begin{array}{lcll} [\![\iota]\!] & = & \mathcal{SN} & \text{if } \iota \text{ is a basic type} \\ [\![A \to B]\!] & = & [\![A]\!] \to [\![B]\!] & \end{array}$$

Remark 20. By Lemma 18, for any type A, $[\![A]\!] \in \mathbf{SAT}$.

Lemma 21. *Let $M, S \in \mathcal{NF}_{\mathcal{L}}$, for any substitution T*

1. *if $M \xrightarrow{\beta_{\mathcal{L}}} M'$, then $(M[T])\!\downarrow_{\mathcal{L}} \xrightarrow{\beta_{\mathcal{L}}} (M'[T])\!\downarrow_{\mathcal{L}}$, and*
2. *if $S \xrightarrow{\beta_{\mathcal{L}}} S'$, then $(S \circ T)\!\downarrow_{\mathcal{L}} \xrightarrow{\beta_{\mathcal{L}}} (S'[T])\!\downarrow_{\mathcal{L}}$.*

Proof. We reason by simultaneous structural induction on M and S. □

Corollary 22. *Let $M, S \in \mathcal{NF}_{\mathcal{L}}$, for any substitution T*

1. *if $(M[T])\!\downarrow_{\mathcal{L}} \in \mathcal{SN}$, then $M \in \mathcal{SN}$, and*
2. *if $(S \circ T)\!\downarrow_{\mathcal{L}} \in \mathcal{SN}$, then $S \in \mathcal{SN}$.*

Lemma 23. *Let $M \in \mathcal{NF}_{\mathcal{L}}$, if for all $N \in [\![A]\!]$, $(M[N \cdot \uparrow^0])\!\downarrow_{\mathcal{L}} \in [\![B]\!]$, then $\lambda_A.M \in [\![A]\!] \to [\![B]\!]$.*

Proof. Let $N \in [\![A]\!]$, we show that $(\lambda_A.M\ N) \in [\![B]\!]$. Since $(\lambda_A.M\ N) \in \mathcal{NT}$, it suffices to prove that if $(\lambda_A.M\ N) \xrightarrow{\beta_{\mathcal{L}}} M''$, then $M'' \in [\![B]\!]$. We have $(M[N \cdot \uparrow^0])\!\downarrow_{\mathcal{L}} \in [\![B]\!] \subseteq \mathcal{SN}$, so by Corollary 22, $M \in \mathcal{SN}$; and by hypothesis, $N \in [\![A]\!] \subseteq \mathcal{SN}$. Thus, we can reason by induction on $\nu(M) + \nu(N)$. In one step $(\lambda_A.M\ N)$ $\beta_{\mathcal{L}}$-reduces to:

- $(M[N \cdot \uparrow^0])\!\downarrow_{\mathcal{L}}$. By hypothesis, $(M[N \cdot \uparrow^0])\!\downarrow_{\mathcal{L}} \in [\![B]\!]$.
- $(\lambda_A.M'\ N)$, with $M \xrightarrow{\beta_{\mathcal{L}}} M'$. By Lemma 21, $(M[N \cdot \uparrow^0])\!\downarrow_{\mathcal{L}} \xrightarrow{\beta_{\mathcal{L}}}$ $(M'[N \cdot \uparrow^0])\!\downarrow_{\mathcal{L}}$. Since, $(M[N \cdot \uparrow^0])\!\downarrow_{\mathcal{L}} \in [\![B]\!]$, we have by Def. 14(2), $(M'[N \cdot \uparrow^0])\!\downarrow_{\mathcal{L}} \in [\![B]\!]$. But also, $\nu(M') < \nu(M)$, so by induction hypothesis, $(\lambda_A.M'\ N) \in [\![B]\!]$.
- $(\lambda_A.M\ N')$, with $N \xrightarrow{\beta_{\mathcal{L}}} N'$. By Def. 14(2), $N' \in [\![A]\!]$, so by hypothesis, $(M[N' \cdot \uparrow^0])\!\downarrow_{\mathcal{L}} \in [\![B]\!]$. But also, $\nu(N') < \nu(N)$, so by induction hypothesis, $(\lambda_A.M\ N') \in [\![B]\!]$.

□

Lemma 24. *For any $\Lambda \in \mathbf{SAT}$, substitution $S \in \mathcal{SN}$, and meta-variable X, $(X[S])\!\downarrow_{\mathcal{L}} \in \Lambda$.*

Proof. Let $M = (X[S])\downarrow_{\mathcal{L}}$, we reason by induction on $\nu(S)$. M is neutral, then by Def. 14(3), it suffices to consider the reductions of M.

- $M \xrightarrow{\beta_{\mathcal{L}}} X$. By Remark 15, $X \in \Lambda$.
- $M \xrightarrow{\beta_{\mathcal{L}}} X[S']$, with $S \xrightarrow{\beta_{\mathcal{L}}} S'$. By hypothesis, $S' \in \mathcal{SN}$ and $\nu(S') < \nu(S)$, so by induction hypothesis, $(X[S'])\downarrow_{\mathcal{L}} = X[S'] \in \Lambda$.

In every case, M $\beta_{\mathcal{L}}$-reduces into terms in Λ, thus by Def. 14(3), $(X[S])\downarrow_{\mathcal{L}} \in \Lambda$. □

Definition 25. The *valuations* of Γ, noted by $[\![\Gamma]\!]$, is a set of substitutions in $\mathcal{NF}_{\mathcal{L}}$ defined inductively on Γ as follows:

$$\begin{array}{lcl} [\![nil]\!] & = & \{\uparrow^n \mid \text{for any natural } n\} \\ [\![A.\Gamma']\!] & = & [\![nil]\!] \cup \{M \cdot S \in \mathcal{NF}_{\mathcal{L}} \mid M \in [\![A]\!], S \in [\![\Gamma']\!]\} \end{array}$$

Notice that if $M \in [\![A]\!]$ and $S \in [\![\Gamma]\!]$, then $M \cdot S$ is not necessarily in $[\![A.\Gamma]\!]$ (since $M \cdot S$ may not be in $\mathcal{NF}_{\mathcal{L}}$). However, we verify easily the following property.

Remark 26. If $M \in [\![A]\!]$ and $S \in [\![\Gamma]\!]$, then $(M \cdot S)\downarrow_{\mathcal{L}} \in [\![A.\Gamma]\!]$.

Lemma 27. *For any Γ, $[\![\Gamma]\!] \subseteq \mathcal{SN}$.*

Proof. We prove by structural induction on S that if $S \in [\![\Gamma]\!]$, then $S \in \mathcal{SN}$.

- $S = \uparrow^n$. In this case S is a $\beta_{\mathcal{L}}$-normal form, then the conclusion is trivial.
- $S = M \cdot T$. By Def. 25, $\Gamma = A.\Gamma'$, $T \in [\![\Gamma']\!]$ and $M \in [\![A]\!] \subseteq \mathcal{SN}$. By induction hypothesis, $T \in \mathcal{SN}$. We prove by induction on $\nu(M) + \nu(T)$ that $M \cdot T \in \mathcal{SN}$ (notice that $M \cdot T \in \mathcal{NF}_{\mathcal{L}}$).

Definition 28. Let $M, S \in \mathcal{NF}_{\mathcal{L}}$, we define

1. Γ satisfies that M is of type A, noted by $\Gamma \models M : A$, if and only if $(M[T])\downarrow_{\mathcal{L}} \in [\![A]\!]$ for any $T \in [\![\Gamma]\!]$.
2. Γ satisfies that S is of type Δ, noted by $\Gamma \models S \rhd \Delta$, if and only if $(S \circ T)\downarrow_{\mathcal{L}} \in [\![\Delta]\!]$ for any $T \in [\![\Gamma]\!]$.

Proposition 29 Soundness of $\models$.

1. *If $\Gamma \vdash M : A$, then $\Gamma \models M : A$,*
2. *If $\Gamma \vdash S \rhd \Delta$, then $\Gamma \models S \rhd \Delta$.*

Proof. By simultaneous induction on derivations $\Gamma \vdash M : A$ and $\Gamma \vdash S \rhd \Delta$. The last applied rule is:

- (Var). In this case, $M = \mathbf{1}$ and $\Gamma = A.\Gamma'$. Let $T \in [\![\Gamma]\!]$, there are three cases:
  - $T = \uparrow^0$. Therefore, $(\mathbf{1}[T])\downarrow_{\mathcal{L}} = \mathbf{1}$. But also, $\mathbf{1}$ is a neutral $\beta_{\mathcal{L}}$-normal form, then by Remark 15, $\mathbf{1} \in [\![A]\!]$.

  - $T = \uparrow^{Suc(n)}$. Therefore, $(\mathbf{1}[T])\downarrow_{\mathcal{L}} = \mathbf{1}[\uparrow^{Suc(n)}]$. But also, $\mathbf{1}[\uparrow^{Suc(n)}]$ is a neutral $\beta_{\mathcal{L}}$-normal form, then by Remark 15, $\mathbf{1}[\uparrow^{Suc(n)}] \in [\![A]\!]$.
  - $T = M' \cdot S'$. Therefore, $(\mathbf{1}[T])\downarrow_{\mathcal{L}} = M'$. By Def. 25 and hypothesis $\Gamma = A.\Gamma'$, we have that $M \in [\![A]\!]$.
- (Clos). In this case $M = M'[S']$, $\Gamma \vdash S \triangleright \Delta$, and $\Delta \vdash M' : A$. We reason by cases analysis on M' and S'.
  - $M' = \mathbf{1}$ and $S' = \uparrow^{Suc(n)}$. Let $T \in [\![\Gamma]\!]$, by induction hypothesis, $(\uparrow^{Suc(n)} \circ T)\downarrow_{\mathcal{L}} \in [\![\Delta]\!]$. Notice that $(\mathbf{1}[\uparrow^{Suc(n)}][T])\downarrow_{\mathcal{L}} = (\mathbf{1}[\uparrow^{Suc(n)} \circ T])\downarrow_{\mathcal{L}}$ $= (\mathbf{1}[(\uparrow^{Suc(n)} \circ T)\downarrow_{\mathcal{L}}])\downarrow_{\mathcal{L}}$. By induction hypothesis, $(\mathbf{1}[(\uparrow^{Suc(n)} \circ T)\downarrow_{\mathcal{L}}])\downarrow_{\mathcal{L}} \in [\![A]\!]$, and thus, $(\mathbf{1}[\uparrow^{Suc(n)}][T])\downarrow_{\mathcal{L}} \in [\![A]\!]$.
  - $M = X$ (X is a meta-variable). Let $T \in [\![\Gamma]\!]$, by induction hypothesis, $(S' \circ T)\downarrow_{\mathcal{L}} \in [\![\Delta]\!]$. Notice that $(X[S'][T])\downarrow_{\mathcal{L}} = (X[S' \circ T])\downarrow_{\mathcal{L}} = (X[(S' \circ T)\downarrow_{\mathcal{L}}])\downarrow_{\mathcal{L}}$. By induction hypothesis, $(X[(S' \circ T)\downarrow_{\mathcal{L}}])\downarrow_{\mathcal{L}} \in [\![A]\!]$, and thus, $(X[S'][T])\downarrow_{\mathcal{L}} \in [\![A]\!]$.
- (Meta$_X$). In this case $M = X$ (X is a meta-variable). Let $T \in [\![\Gamma]\!]$, there are two cases:
  - $T = \uparrow^0$. Therefore, $(X[T])\downarrow_{\mathcal{L}} = X$. But also, X is a neutral $\beta_{\mathcal{L}}$-normal form, then by Remark 15, $X \in [\![A]\!]$.
  - $T \neq \uparrow^0$. Therefore, $(X[T])\downarrow_{\mathcal{L}} = X[T]$. By Lemma 27, $T \in \mathcal{SN}$, then by Lemma 24, $X[T] \in [\![A]\!]$.
- (Abs). In this case $M = \lambda_{A_1}.M_1$, $A_1.\Gamma \vdash M_1 : B_1$, and $A = A_1 \to B_1$. By Def. 19, $[\![A]\!] = [\![A_1 \to B_1]\!] = [\![A_1]\!] \to [\![B_1]\!]$. Let $T \in [\![\Gamma]\!]$ and $\Uparrow(T)$ be a notation for $\mathbf{1} \cdot (T \circ \uparrow^{Suc(0)})$. We have $((\lambda_{A_1}.M_1)[T])\downarrow_{\mathcal{L}} = \lambda_{A_1}.(M_1[\Uparrow(T)])\downarrow_{\mathcal{L}}$. By Lemma 23, it suffices to prove that for any $N \in [\![A_1]\!]$, $((M_1[\Uparrow(T)])\downarrow_{\mathcal{L}}[N \cdot \uparrow^0])\downarrow_{\mathcal{L}} \in [\![B_1]\!]$. By hypothesis and Remark 26, $(N \cdot T)\downarrow_{\mathcal{L}} \in [\![A_1.\Gamma]\!]$, then by induction hypothesis, $(M[(N \cdot T)\downarrow_{\mathcal{L}}])\downarrow_{\mathcal{L}} =$ $((M_1[\Uparrow(T)])\downarrow_{\mathcal{L}}[N \cdot \uparrow^0])\downarrow_{\mathcal{L}} \in [\![B_1]\!]$.
- (App). In this case $M = (M_1\ N_1)$, $\Gamma \vdash M_1 : B \to A$ and $\Gamma \vdash N_1 : B$. Let $T \in [\![\Gamma]\!]$, so we have, $((M_1\ N_1)[T])\downarrow_{\mathcal{L}} = ((M_1[T])\downarrow_{\mathcal{L}}\ (N_1[T])\downarrow_{\mathcal{L}})$. By induction hypothesis, $(M_1[T])\downarrow_{\mathcal{L}} \in [\![B \to A]\!] = [\![B]\!] \to [\![A]\!]$ and $(N_1[T])\downarrow_{\mathcal{L}} \in [\![B]\!]$. Hence, $((M_1\ N_1)[T])\downarrow_{\mathcal{L}} \in [\![A]\!]$.
- (Id), (Shift). In this case $S = \uparrow^n$. We prove by structural induction on n and T that if $T \in [\![\Gamma]\!]$ and $\Gamma \vdash \uparrow^n \triangleright \Delta$, then $(\uparrow^n \circ T)\downarrow_{\mathcal{L}} \in [\![\Delta]\!]$.
- (Cons). In this case $S = M' \cdot S'$, $\Gamma \vdash M' : A'$, $\Gamma \vdash S' \triangleright \Delta'$ and $A'.\Delta' = \Delta$. Let $T \in [\![\Gamma]\!]$, so we have, $(S \circ T)\downarrow_{\mathcal{L}} = ((M'[T])\downarrow_{\mathcal{L}} \cdot (S' \circ T)\downarrow_{\mathcal{L}})\downarrow_{\mathcal{L}}$. By induction hypothesis, $(M'[T])\downarrow_{\mathcal{L}} \in [\![A']\!]$ and $(S' \circ T)\downarrow_{\mathcal{L}} \in [\![\Delta']\!]$. From Remark 26 we conclude that $((M'[T])\downarrow_{\mathcal{L}} \cdot (S' \circ T)\downarrow_{\mathcal{L}})\downarrow_{\mathcal{L}} \in [\![\Delta]\!]$.

$\square$

Theorem 30. *Let M, S be expressions in $\mathcal{NF}_{\mathcal{L}}$.*

1. *If $\Gamma \vdash M : A$, then $M \in \mathcal{SN}$.*
2. *If $\Gamma \vdash S \triangleright \Delta$, then $S \in \mathcal{SN}$.*

Proof. By Def. 25, $\uparrow^0 \in [\![\Gamma]\!]$. Hence,

1. By Proposition 29, $(M[\uparrow^0])\downarrow_{\mathcal{L}} = M \in [\![A]\!]$, and by Def. 14, $[\![A]\!] \subseteq \mathcal{SN}$.

2. By Proposition 29, $(S \circ \uparrow^0){\downarrow_{\mathcal{L}}} = S \in [\![\Delta]\!]$, and by Lemma 27, $[\![\Delta]\!] \subseteq \mathcal{SN}$.

□

Theorem 31. *If $\Gamma \vdash M : A$ and $\Gamma \vdash S \rhd \Delta$, then M and S are weakly normalizing, and thus M and S have $\lambda_{\mathcal{L}}$-normal forms.*

Proof. Let $N = M{\downarrow_{\mathcal{L}}}$ and $T = S{\downarrow_{\mathcal{L}}}$, the subject reduction property (Theorem 10) says that typing is preserved under reductions, hence $\Gamma \vdash N : A$ and $\Gamma \vdash T \rhd \Delta$. Therefore, by Theorem 30, N and T are both in $\mathcal{SN}$. Finally, remark that a $\beta_{\mathcal{L}}$-normal form in $\mathcal{NF}_{\mathcal{L}}$ is a $\lambda_{\mathcal{L}}$-normal form too. □

6 Conclusions

We have proposed a variant of $\lambda\sigma$, namely $\lambda_{\mathcal{L}}$. This calculus enjoys the same general properties of $\lambda\sigma$:

– a simple and finitary first-order rewrite system,
– confluent on terms with meta-variables,
– weakly terminating on typed terms and
– with composition of substitutions and simultaneous substitutions.

However, in contrast to $\lambda\sigma$, $\lambda_{\mathcal{L}}$ does not have the (SCons)-rule and so, it is left-linear in the sort of terms and substitutions.

Although $\lambda_{\mathcal{L}}$ was designed to allow meta-variables, it happens to be useful in the same framework where $\lambda\sigma$ is. In particular both calculi share the same description of normal forms. For example, the higher-order unification algorithm via explicit substitutions proposed in [4] can be expressed in $\lambda_{\mathcal{L}}$, almost without modifications. Moreover, since $\lambda_{\mathcal{L}}$ does not have the surjective pairing rule, it is useful for applications where this feature of $\lambda\sigma$ pose technical problems, for instance higher-order equational unification via explicit substitutions [19], or dependent type systems [27].

Another left-linear variant of $\lambda\sigma$ is the $\lambda_{\Uparrow}$-calculus [3]. The system $\lambda_{\Uparrow}$ is fully confluent on open terms, not only with meta-variables of terms but also with meta-variables of substitutions. However, $\lambda_{\Uparrow}$ is incompatible with the extensional rule (η) due to the fact that substitutions id and $\mathbf{1}\cdot\uparrow$ are not $\lambda_{\Uparrow}$-convertible. A key point in $\lambda_{\mathcal{L}}$ is the preservation of this extensional equivalence. The extensional version of $\lambda_{\mathcal{L}}$-calculus is confluent on ground terms as shown in [18], and we conjecture that it is also on semi-open expressions.

The $\lambda_{\mathcal{L}}$-calculus is extended to dependent types in [27] and work is in progress to use this calculus in a formulation of the Calculus of Inductive Constructions with explicit substitutions and open expressions.

Acknowledgments Many thanks to all persons contributing to this work, in particular to Gilles Dowek, Delia Kesner, Benjamin Werner, Bruno Barras, Gopalan Nadathur and the anonymous referees for their useful remarks and suggestions on the subject of this paper. The author is very grateful with Thomas Arts and Hans Zantema for their help with Proposition 1, in particular Hans Zantema send to me a proof of this proposition in a personal communication [36].

References

1. M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit substitution. *Journal of Functional Programming*, 1(4):375–416, 1991.
2. R. Bloo and K. H. Rose. Preservation of strong normalisation in named lambda calculi with explicit substitution and garbage collection. In *CSN-95: Computer Science in the Netherlands*, November 1995.
3. P.-L. Curien, T. Hardin, and J.-J. Lévy. Confluence properties of weak and strong calculi of explicit substitutions. *Journal of the ACM*, 43(2):362–397, March 1996.
4. 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.
5. G. Dowek, T. Hardin, C. Kirchner, and F. Pfenning. Unification via explicit substitutions: The case of higher-order patterns. In M. Maher, editor, *Proceedings of the Joint International Conference and Symposium on Logic Programming*, Bonn, Germany, September 1996. MIT Press. To appear.
6. M. C. F. Ferreira, D. Kesner, and L. Puel. λ-calculi with explicit substitutions and composition which preserve β-strong normalization. *LNCS*, 1139, 1996.
7. H. Geuvers. A short and flexible proof of Strong Normalization for the Calculus of Constructions. In P. Dybjer and B. Nordström, editors, *Types for Proofs and Programs, International Workshop TYPES'94*, volume 996 of *LNCS*, pages 14–38, Båstad, Sweden, 1994. Springer.
8. J.-Y. Girard, P. Taylor, and Y. Lafont. *Proof and Types*. Cambridge University Press, 1989.
9. J. Goubault-Larrecq. A proof of weak termination of typed $\lambda\sigma$-calculi. Manuscript, 1997.
10. J. Goubault-Larrecq. Une preuve de terminaison faible du $\lambda\sigma$-calcul. Technical Report RR-3090, Unité de recherche INRIA-Rocquencourt, Janvier 1997.
11. T. Hardin. Confluence results for the Pure Strong Categorical Logic CCC: λ-calculi as subsystems of CCL. *Theoretical Computer Science*, 65(2):291–342, 1989.
12. 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.
13. G. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *J.A.C.M.*, 27(4), October 1980.
14. F. Kamareddine and A. Rios. A λ-calculus à la de Bruijn with explicit substitutions. In *PLILP*. LNCS, 1995.
15. D. Kapur, P. Narendran, and F. Otto. On ground-confluence of term rewriting systems. *Information and Computation*, 86(1):14–31, May 1990.
16. 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.
17. D. Kesner. Confluence properties of extensional and non-extensional λ-calculi with explicit substitutions (extended abstract). In Harald Ganzinger, editor, *Proceedings of the 7th International Conference on Rewriting Techniques and Applications (RTA-96)*, volume 1103 of *LNCS*, pages 184–199, New Brunswick, NJ, USA, 1996. Springer-Verlag.
18. D. Kesner. Confluence of extensional and non-extensional λ-calculi with explicit substitutions. Preprint, 1997.

19. C. Kirchner and C. Ringeissen. Higher order equational unification via explicit substitutions. Preprint, 1996.
20. 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.
21. P. Lescanne and J. Rouyer-Degli. Explicit substitutions with de Bruijn's levels. In J. Hsiang, editor, *Rewriting Techniques and Applications*, volume 914 of *LNCS*, pages 294–308, Chapel Hill, North Carolina, 1995. Springer-Verlag.
22. L. Magnusson. *The Implementation of ALF—A Proof Editor Based on Martin-Löf's Monomorphic Type Theory with Explicit Substitution.* PhD thesis, Chalmers University of Technology and Göteborg University, January 1995.
23. P.-A. Melliès. Exemple de non terminaison forte dans un $\lambda\sigma$-calcul typé où la priorité serait donnée aux deux règles *shiftcons* et *varcons*, modulo lois de monoïde. Preprint, 1995.
24. 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.
25. C. Muñoz. Confluence and preservation of strong normalisation in an explicit substitutions calculus (extended abstract). In *Proceedings, Eleven Annual IEEE Symposium on Logic in Computer Science*, New Brunswick, New Jersey, July 1996. IEEE Computer Society Press.
26. C. Muñoz. Proof representation in type theory: State of the art. In *Proceedings, XXII Latinamerican Conference of Informatics CLEI Panel 96*, Santafé de Bogotá, Colombia, June 1996.
27. C. Muñoz. Dependent types with explicit substitutions: A meta-theoretical development. Preprint electronically available at: `http://pauillac.inria.fr/~cesar/Papers/typ96.ps.gz`, 1997.
28. C. Muñoz. Meta-theoretical properties of λ_ϕ: A left-linear variant of $\lambda\sigma$. Technical Report RR-3107, Unité de recherche INRIA-Rocquencourt, Février 1997.
29. G. Nadathur. A fine-grained notation for lambda terms and its use in intensional operations. Technical Report TR-96-13, Department of Computer Science, University of Chicago, May 30 1996.
30. G. Nadathur. The (SCons) rule. Personal communication, 1996.
31. B. Pagano. Confluent extensions of $\lambda_\Uparrow$. Personal communication, 1996.
32. A. Ríos. *Contributions à l'étude de λ-calculs avec des substitutions explicites.* PhD thesis, U. Paris VII, 1993.
33. M. Schmidt-Schauss. *Computational aspects of an order-sorted logic with term declarations*, volume 395 of *Lecture Notes in Computer Science and Lecture Notes in Artificial Intelligence.* Springer-Verlag Inc., New York, NY, USA, 1989.
34. H. Yokouchi and T. Hikita. A rewriting system for categorical combinators with multiple arguments. *SIAM Journal on Computing*, 19(1):78–97, February 1990.
35. H. Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24:89–105, 1995.
36. H. Zantema. Termination of ϕ and Π_ϕ by semantic labelling. Personal communication, 1996.

This article was processed using the LaTeX macro package with LLNCS style