Available online at www.sciencedirect.com
SCIENCE DIRECT
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 123 (2005) 213–228
www.elsevier.com/locate/entcs

Abstract Conditions for the Confluence of Explicit Substitution Calculi

Amelia Râştei

ULTRA group, School of Mathematical and Computational Sc., Heriot-Watt University, Edinburgh, UK, email: amelia@macs.hw.ac.uk

Abstract

The aim of this paper is to give abstract properties of some calculi with explicit substitution which will be sufficient to prove their confluence. We define a property that we call "implementing a good notion of substitution." We show that calculi with explicit substitution having this property are confluent and their substitution reductions are also confluent. We test our theory with the well-known calculi of explicit substitution $\lambda s, \lambda \upsilon$ and λse. The latter is λs with open terms.
The property of implementing a good substitution is natural and characterizes a large number of calculi. Two conditions of this property are usually checked as an initial step in the proof for confluence. The third condition is new and is the key for our proofs of confluence.

Keywords: Lambda calculi, explicit substitution, β-reduction, confluence, good substitution

1 The common ground of calculi of explicit substitutions

In the last ten-fifteen years many calculi of explicit substitution have been proposed. Calculi of explicit substitution are based on the λ-calculus and aim at controlling the process of substitution. In the calculi of explicit substitution, the operation of substitution is performed in a stepwise way via, what we call here, substitution reduction steps. A fundamental property of these calculi is confluence: confluence of the substitution reduction and confluence of the β-reduction, the latter implying the former. Confluence is equivalent to saying that the interconvertibility of two terms can be checked by converting them to a common form. Confluence can be an involved property to prove for some calculi (see, for example, the proof for the confluence of λse in [12]). This

1571-0661
doi:10.1016/j.entcs.2004.05.016

paper provides a natural approach for proving confluence for calculi which can be shown to have the so-called *implementing a good substitution* property.

To build our approach, we start from the common ground of calculi with explicit substitution: the λ-calculus. There are two main styles of building calculi of explicit substitution from the λ-calculus: $\lambda\sigma$-style [1,14] and λs-style [12]. Some relations between the two styles are given in [13]. For a calculus in either styles, the set of rules consists of one rule that starts the simulation of the β-reduction in the λ-calculus, which we call simply β- reduction, and a number of substitution reduction rules.

A standard method for showing confluence of explicit substitution calculi is the interpretation method, that 'maps' the calculus of explicit substitution into an *interpretation calculus* that is known to be confluent. For calculi without open terms, the interpretation calculus is λ-calculus, see, for example, the proofs for confluence of λs [11] and $\lambda\upsilon$ [2]. For calculi with open terms, the interpretation calculus is a middle-calculus of substitution normal forms, this is the case, for example, for λse [12] and λ_{ws} [7]. The method we give in this paper, however, uses λ-calculus as an interpretation calculus for λse. We do not treat in this paper the calculi λ_{ws} and $\lambda\sigma$ and hope that our results can be generalized in the future to these calculi too. When dealing with the calculi we consider, two main properties of the interpretation calculus are necessary: confluence of the β-reduction and preservation of β-reduction by the operation of substitution. When the interpretation calculus is the λ-calculus, these properties are obtained for free, provided that the operation of substitution in that calculus is "good" or correctly defined. We explain briefly what we mean by this.

In the λ-calculus, the definition of the β-reduction is based on the notion of substitution. The confluence of the β-reduction means that the final chain of substitutions does not depend on the order in which substitutions are performed. This means that confluence of β-reduction depends on some "good" properties of substitution. Substitution is not simple replacement. Careful attention is needed when defining substitution because of the binding structure of the λ-calculus, generated by the binding operator denoted by λ. The operator λ binds a variable inside a term, so variables can occur free or bound in a term. The definition of free and bound occurrences of variables is as usual. In order to be correctly defined, substitution must preserve the status of the free and bound variables. This informal requirement is often called "avoiding clashes of variables" or "avoiding captures of variables" or "avoiding confusion of bound variables" and intuitively is well understood by those working with λ-calculus based systems. Some examples of notions of substitution that satisfy this requirement are Curry and Feys substitution [4],

the simultaneous substitution of Stoughton [17], the substitution for the λ-calculus with de Bruijn indices [8] and its variants, e.g., the two substitutions used in this paper. All these notions of substitution have in common not only the property of "avoiding clashes of variables", but also properties like the Substitution Lemma, preservation of β-reduction. Moreover, the β-reduction defined with such a notion of substitution is confluent. We will call such substitutions "good substitutions". The property of a substitution of being "good" does not depend on the particular encoding that is used for the syntax of the λ-calculus. Some of the examples of good substitutions we gave above are based on an encoding with variable names, some on an encoding with indices. The length of this paper does not allow us to give a precise definition and to develop a theory of good substitutions in the sense described above. We do this in an unpublished report [1]. In this paper we work with the λ-calculus encoded with de Bruijn indices and with two notions of substitutions that are already known to be "good" and often used in the literature. Our results would hold if one considers any encoding of the λ-calculus and any good notion of substitution.

Yet another approach to defining good substitutions and, in the same time, control the evaluation by β-reduction steps, is the one of calculi of explicit substitution [1,2,3,10,11,14,6,15,16]. We exploit, in this paper, the abstract rules by which they have been built from λ-calculus. We show that a calculus that "implements a good notion of substitution" is confluent.

We summarize the results of this papers as follows.

1. We show that a calculus of explicit substitution that "implements a good notion of substitution", is confluent.
2. We show that $\lambda s, \lambda \upsilon$ and λse are confluent using our method.
3. We compare the difficult part and easy parts of the original proof for λse and our proof.
4. We discuss the feasibility of applying our method to other calculi.

The structure of the rest of the paper is as follows. In Section 2 introduces the λ-calculus with de Bruijn indices as used in the literature, proofs of the meta-properties are not included in the article, but are found in the literature (see, for example, [12]). In Section 3 we develop our theory for testing explicit substitution calculi. The remaining sections will apply our method to λs and $\lambda \upsilon$ and finally, to λs with open terms. In the end of the paper we conclude with a discussion on what has been achieved, the relevance of this work and directions for future work.

[1] "Good substitutions in the λ-calculus", http://www.macs.hw.ac.uk/~amelia/

2 The λ-calculus with de Bruijn indices

Let $(\Lambda_{\mathrm{DB}}, \longrightarrow_{\beta(S)}, \mathsf{Eq}_S)$ be the rewriting system of the λ-calculus with de Bruijn indices. The set of terms, Λ_{DB}, the β-rule, $\longrightarrow_{\beta(S)}$, parametrized with a substitution S and the equations Eq_S for defining the substitution S are given below. We often do not mention the set of equations for defining substitution, and write $(\Lambda_{\mathrm{DB}}, \longrightarrow_{\beta(S)})$ instead of $(\Lambda_{\mathrm{DB}}, \longrightarrow_{\beta(S)}, \mathsf{Eq}_S)$. The set of terms Λ_{DB} is defined by the grammar:

$$\Lambda_{\mathrm{DB}} \ ::= \ \mathbb{N} \mid (\Lambda_{\mathrm{DB}}\Lambda_{\mathrm{DB}}) \mid \lambda\Lambda_{\mathrm{DB}}$$

Let M, N, P range over Λ_{DB} and $\mathbb{N} = \{1, 2, 3, \ldots\}$. The notion of substitution with global updating together with the operations for updating the indices are given below. Let S be a good notion of substitution, then the β-reduction rule is defined as follows ($S \in \{S_{\mathsf{b}}, S_{\mathsf{t}}\}$):

$$(\lambda M)N \rightarrow_{\beta(S)} S(N, 1, M).$$

Definition 2.1 The equations of $\mathsf{Eq}_{S_{\mathsf{b}}}$ for defining the substitution with global updating of the λ-calculus with de Bruijn indices are:

$$S_{\mathsf{b}}(N,k,M_1M_2) = S_{\mathsf{b}}(N,k,M_1)S_{\mathsf{b}}(N,k,M_2) \qquad U_i^k(M_1M_2) = U_i^k(M_1)U_i^k(M_2)$$
$$S_{\mathsf{b}}(N,k,\lambda M) = \lambda S_{\mathsf{b}}(N,k+1,M) \qquad U_i^k(\lambda M) = \lambda U_{i+1}^k(M)$$
$$S_{\mathsf{b}}(N,k,m) = \begin{cases} m-1 & \text{if } m > k \\ U_0^k(N) & \text{if } m = k \\ m & \text{if } m < k \end{cases} \qquad U_i^k(m) = \begin{cases} m+k-1 & \text{if } m > i \\ m & \text{if } m \leq i \end{cases}$$

Definition 2.2 The equations of $\mathsf{Eq}_{S_{\mathsf{t}}}$ for defining the substitution with local updating in the λ-calculus with de Bruijn indices are:

$$S_{\mathsf{t}}(N,k,M_1M_2) = S_{\mathsf{t}}(N,k,M_1)S_{\mathsf{t}}(N,k,M_2) \qquad u_i(M_1M_2) = u_i(M_1)u_i(M_2)$$
$$S_{\mathsf{t}}(N,k,\lambda M) = \lambda S_{\mathsf{t}}(u_0(N),k+1,M) \qquad u_i(\lambda M) = \lambda u_{i+1}(M)$$
$$S_{\mathsf{t}}(N,k,m) = \begin{cases} m-1 & \text{if } m > k \\ N & \text{if } m = k \\ m & \text{if } m < k \end{cases} \qquad u_i(m) = \begin{cases} m+1 & \text{if } m > i \\ m & \text{if } m \leq i \end{cases}$$

By orienting the equations of Eq_S with $S \in \{S_{\mathsf{b}}, S_{\mathsf{t}}\}$, from left to right, they can be treated as a rewriting system. We denote this orientation with the symbol $\succ$ instead of "=". We use the notation $\mathsf{Eq}_{S,\succ}$ for the rewriting rules thus obtained and $=_{\mathsf{Eq}_S}$ for the reflexive, symmetric, transitive closure of $\succ$. The rewriting system thus obtained is terminating and confluent. One can easily show termination, using the recursive path ordering method of Derschowitz or/and the lexicographic path ordering method of Kamin and Lévy (methods described in [5]). Since $\succ$ is terminating and normal forms are unique we conclude that $\succ$ is confluent.

The notion of substitutions $S_{\mathsf{b}}, S_{\mathsf{t}}$ are good, i.e., avoid clashes of variables. Two properties are of special importance for our results: Theorem 2.3 and Theorem 2.4.

Theorem 2.3 *Let $S \in \{S_{\mathsf{b}}, S_{\mathsf{t}}\}$. Then, S preserves $\longrightarrow_{\beta(S)}$.*

By preserving β-reduction we mean the following: Let S be any good notion of substitution and $M \longrightarrow_{\beta(S)} N$. Then, for any term Q and variable k we have $S(Q,k,M) \longrightarrow_{\beta(S)} S(Q,k,N)$ and $S(M,k,Q) \xrightarrow{*}_{\beta(S)} S(N,k,Q)$.

Theorem 2.4 *The λ-calculus $(\Lambda_{\mathrm{DB}}, \longrightarrow_{\beta(S)})$, where $S \in \{S_{\mathsf{b}}, S_{\mathsf{t}}\}$ is confluent.*

The properties of substitution S_{b} and updating are expressed by the following lemmas (cf. [12]). Similar properties can be established for S_{t}. For the λ-calculus with no meta-variables over terms, the equalities of the lemma given below can be obtained from the equations (or rules) defining S_{b}. One says that, in this case, the equations below are "admissible".

Lemma 2.5 *Let X, Y, Z be a meta-variables ranging over terms. Then, the following equalities hold.*

1. *For $k < n < k+i$ we have: $S_{\mathsf{b}}(X, n, U_k^i(Y)) = U_k^{i-1}(Y)$.*
2. *For $l \leq k < l+j$ we have: $U_k^i(U_l^j(X)) = U_l^{j+i-1}(X)$.*
3. *For $k+i \leq n$ we have: $S_{\mathsf{b}}(Y, n, U_k^i(X)) = U_k^i(S_{\mathsf{b}}(Y, n-i+1, X))$.*
4. *For $i \leq n$ we have:*

$S_{\mathsf{b}}(Z, n, S_{\mathsf{b}}(Y, i, X)) = S_{\mathsf{b}}(S_{\mathsf{b}}(Z, n-i+1, Y), i, S_{\mathsf{b}}(Z, n+1, X))$.

5. *For $l+j \leq k+1$ we have: $U_k^i(U_l^j(X)) = U_l^j(U_{k+1-j}^i(X))$.*
6. *For $n \leq k+1$ we have: $U_k^i(S_{\mathsf{b}}(Y, n, X)) = S_{\mathsf{b}}(U_{k-n+1}^i(Y), n, U_{k+1}^i(X))$.*

In the case of λ-terms extended with meta-variables, the equations of Lemma 2.5 are not admissible anymore. Therefore, we extend $\mathsf{Eq}_{S_{\mathsf{b}},\succ}$ such that $\succ$ orients the equations of Lemma 2.5 from left to right. The extended $\succ$ is weakly terminating: one can define an innermost terminating strategy by postponing the use of rules corresponding to meta-properties of S_{b} after all the possible rules in $\mathsf{Eq}_{S_{\mathsf{b}}}$ are applied. Moreover, $\succ$ is confluent: the normal forms are unique and $\succ$ is weakly terminating. Confluence can also be established noticing that the oriented equations from left ro right of Lemma 2.5 are left linear and with no critical pairs. When considering explicit substitution calculi with open terms (i.e., terms containing meta-variables) that implements a good substitution S, we extended the set $\mathsf{Eq}_{S,\succ}$ with oriented equalities corresponding to the meta-properties of S.

3 Criteria: implementing a good notion of substitution

Let $(\Lambda_{\mathrm{ES}}, \longrightarrow_{\mathrm{Beta}}, \mathsf{Red}_S)$ be a general calculus of explicit substitution. The set Λ_{ES} contains pure terms and those containing substitutions, $\longrightarrow_{\mathrm{Beta}}$ is the β-reduction generating a term containing a substitution and Red_S is the set of reduction rules that propagate and eliminate substitutions. Such calculus is

obtained from $(\Lambda_{\mathrm{DB}}, \longrightarrow_{\beta(S)}, \mathsf{Eq}_{S,\succ})$ by making substitution an explicit operator. The β-reduction rule for the λ-calculus is now split into two reductions: $\longrightarrow_{\mathrm{Beta}}$, and the reduction rules of Red_S. The subcalculus $(\Lambda_{\mathrm{ES}}, \mathsf{Red}_S)$ is called the substitution subcalculus. We introduce now our criteria and then we justify it.

Definition 3.1 Assume $(\Lambda_{\mathrm{DB}}, \longrightarrow_{\beta(S)}, \mathsf{Eq}_{S,\succ})$, where S is a good substitution and $\succ$ is a confluent and weakly-terminating ordering of equations in Eq_S. We say that $(\Lambda_{\mathrm{ES}}, \longrightarrow_{\mathrm{Beta}}, \mathsf{Red}_S)$ implements S iff there $\exists T : \Lambda_{\mathrm{ES}} \to \Lambda_{\mathrm{DB}}$ such that:

(BET) $M \longrightarrow_{\beta(S)} N$ iff $\exists N'$ such that $M \longrightarrow_{\mathrm{Beta}} N'$ and $T(N') =_{\mathsf{Eq}_S} N$, $M, N \in \Lambda_{\mathrm{DB}}$ and $N' \in \Lambda_{\mathrm{ES}}$.

(GS1) If $A \to B$, then $\exists k \geq 1$ finite such that $T(A) \succ^k T(B)$.

(GS2) If $T(A) \succ T(B)$, then $\exists k \geq 1$ finite such that $A \xrightarrow{k} B$, $A, B \in \Lambda_{\mathrm{ES}}$.

The function T is a total function and is usually called the "interpretation function". It is common to define and use an interpretation function for showing the confluence of the substitution subcalculus and of the whole explicit calculus. This is the case of several calculi: $\lambda s, \lambda t$ [12], $\lambda \upsilon$ [2], $\lambda \zeta$ [10]. The novelty in the definition above is the condition (GS2). It is this condition that will help us simplify the proof for confluence of the explicit substitution calculi.

Before we go further we should ask ourselves whether the three conditions are legitimate, in the sense that they are desirable and characterize as many calculi as possible. The first condition, (BET), illustrates the splitting of a $\longrightarrow_{\beta(S)}$ step into a step $\longrightarrow_{\mathrm{Beta}}$ followed by some other reductions that must simulates the operation of substitution generated by the $\longrightarrow_{\beta(S)}$ step. Conditions (GS1) and (GS2) describe a harmonious correspondence between a $\succ$-step and a substitution reduction step $\longrightarrow$. A substitution reduction step is formed in agreement with a $\succ$-step. (GS1) states the fact that no "new" $\longrightarrow$ steps are formed in the explicit substitution calculus, but all must correspond to some $\succ$-steps in the λ-calculus. (GS2) states that no $\succ$-step is "annihilated" in the explicit substitution calculus, but corresponds to a finite number of substitution reduction steps. Under the hypothesis of our criteria, both (GS1) and (GS2) are necessary to prove confluence of the substitution reduction.

Lemma 3.2 *A calculus of explicit substitution that implements a good substitution is weakly terminating.*

Proof. The function T describes a strategy: applying a finite number of $\succ$ steps to a term $T(M), M \in \Lambda_{\mathrm{ES}}$ one obtains a term in Λ_{DB}. Since $\succ$ is

weakly terminating and (GS2) holds, the substitution reduction is also weakly terminating. □

Theorem 3.3 *The substitution subcalculus of a calculus of explicit substitution that implements a good notion of substitution is confluent.*

Proof. Let $M \xrightarrow{*} N$ and $M \xrightarrow{*} P$ be any two sequences of reductions in Red_S. From (GS1) we have $T(M) \succ^* T(N)$ and $T(M) \succ^* T(P)$. Also, $\succ$ is confluent. Using now (GS2) we obtain the conclusion. □

From the conditions (GS1) and (GS2) we cannot conclude that the substitution subcalculus is terminating, but merely weak-terminating. Also, the condition (BET) does not require strong termination of the substitution reduction, but weakly termination is necessary and sufficient. Strong normalization is quite difficult to show in many cases and known to be undecidable. Calculi known to be strong confluent are, for example, $\lambda\sigma$, $\lambda s, \lambda t, \lambda \upsilon, \lambda\zeta$.

Confluence

Usually, the proof of the confluence of the explicit substitution calculi is based on the interpretation method [9], where the termination of the substitution subcalculus is a requirement. We use the generalized method, called short GIM, of [12] that is suitable for showing the confluence of explicit substitution calculi when strong normalization cannot be established.

Theorem 3.4 *Let $R = R_1 \cup R_2$, where R_1, R_2 are arbitrary reductions on a set A. Let B be a set of R_1-normal forms and let $T : A \to B$ be a function such that $T(M)$ is a R_1-normal form of M. If there exists a reduction R' on the set of R_1-normal forms satisfying*
1. $R' \subseteq R^*$
2. $\forall M, N : M \to_{R_1} N \Longrightarrow T(M) \xrightarrow{*}_{R'} T(N)$.
3. $\forall M, N : M \to_{R_2} N \Longrightarrow T(M) \xrightarrow{*}_{R'} T(N)$.
then R' is confluent iff R is confluent.

Note that R^* is the reflexive, transitive closure of a relation R. We show that if a calculus with explicit substitution implements a good substitution, then the conditions of GIM hold, so the calculus is confluent.

Theorem 3.5 *A calculus with explicit substitution that implements a good notion of substitution is confluent.*

Proof. In Theorem 3.4 we take R' to be the β-reduction in the λ-calculus with de Bruijn indices, R_1 to be the substitution reduction and R_2 to be the β-reduction in the calculus of explicit substitution. The condition 1. of Theorem 3.4 is satisfied because of the conditions (BET) and Theorem 3.3.

From (BET) we know that if $M \longrightarrow_{\beta(S)} N$ then there exists N' such that $M \longrightarrow_{\text{Beta}} N'$ and $T(N') =_{\mathsf{Eq}_S} N$. Using (GS2) we can easily notice that there exists k such that $N' \overset{k}{\longleftrightarrow} N$. Using now Theorem 3.3, we know that the substitution reduction is confluent, hence Church-Rosser. So there exists l such that $N' \xrightarrow{l} N$, since N is a normal form with respect to the substitution reduction. The condition 2. is also satisfied (see condition GS1). To show that 3. holds, we use the property of a good substitution of preserving β-reduction steps (Theorem 2.3). The case when $M \in \Lambda_{\text{DB}}$ is straightforward from (BET). We take now the remaining case. Assume that a substitution term is denoted by $M\sigma$, where σ is a substitution, i.e. a function with a finite domain from the set of variables to the set of terms. The domains of S_{b} and S_{t}, for example, contain only one element. The term $M\sigma$ is translated into a term $\sigma'(T(M))$ where $\forall i.\ \sigma'(i) = T(\sigma(i))$. We use the fact that a good substitution σ preserves β-reduction steps. If $M \longrightarrow_{\text{Beta}} M_1$ and/or $\sigma \longrightarrow_{\text{Beta}} \sigma_1$ (i.e., $\exists i.\ \sigma(i) \longrightarrow_{\text{Beta}} \sigma_1(i)$ and $\forall j \neq i.\ \sigma(j) = \sigma_1(j)$), then by induction hypothesis we have that $\sigma'(T(M)) \xrightarrow{*}_{\beta(\sigma)} \sigma'_1(T(M_1))$, where $\forall i.\ \sigma'_1(i) = T(\sigma_1(i))$. □

In the next sections we apply our criteria for showing confluence to diverse explicit substitution calculi: λs and $\lambda \upsilon$, the latter being in the $\lambda\sigma$-style. We apply our method for the explicit substitution calculus λs extended with open terms.

4 Calculi with explicit substitution in λs-style

The calculus λs is an explicit substitution calculus with de Bruijn indices. The set of terms and rules is given below. The rules of the λs calculus are such that all operations of lifting of indices are postponed after the propagation of substitution operations. This postponement is sometimes called "global updating". A similar calculus with λs is λt which has "local updating" because it allows steps of propagating substitutions to be interleaved with operation of lifting. The calculus λt is based on the substitution S_{t}.
The set of terms Λs of λs is given as follows.

$$\Lambda s \ ::= \ \mathbb{N} \mid (\Lambda s \Lambda s) \mid (\lambda \Lambda s) \mid (\Lambda s \sigma^i \Lambda s) \mid (\varphi^k_j \Lambda s) \text{ where } i, k \geq 1, j \geq 0.$$

Let $M, N, P, \ldots$ range over Λs. Figure 1 gives the rules of λs. All the rules except σ-generation, propagate substitutions to the variable level where they are finally performed. In fact, λs was obtained from the λ-calculus with de Bruijn indices via two steps: (1) the substitution operation S_{b} together with the updating operator U become the explicit operators σ and φ respectively, and terms built via these operators are added to the set of terms; (2) all the equations defining the substitution operation S_{b} become oriented reduction

σ-generation	$(\lambda M)N \to M\sigma^1 N$
σ-λ-transition	$(\lambda M)\sigma^j N \to \lambda(M\sigma^{j+1}N)$
σ-app-transition	$(M_1M_2)\sigma^j N \to (M_1\sigma^j N)(M_2\sigma^j N)$
σ-destruction	$n\sigma^j N \to \begin{cases} n-1 & \text{if } n > j \\ \varphi_0^j N & \text{if } n = j \\ n & \text{if } n < j \end{cases}$
φ-λ-transition	$\varphi_k^i(\lambda M) \to (\lambda\varphi_{k+1}^i M)$
φ-app-transition	$\varphi_k^i(M_1M_2) \to (\varphi_k^i M_1)(\varphi_k^i M_2)$
φ-destruction	$\varphi_k^i n \to \begin{cases} n+i-1 & \text{if } n > k \\ n & \text{if } n \le k \end{cases}$

Fig. 1. Rules of λs.

rules. The sub-calculus obtained from λs by removing the rule σ-generation is called the substitution subcalculus.

Lemma 4.1 *λs implements the good notion of substitution S_{b}.*

Proof. The proof is by constructing the function T as follows.
$-T(MN) = T(M)T(N)$ $\quad -T(\lambda M) = \lambda T(M)$ $\quad -T(n) = n$
$-T(M\sigma^k N) = S_{\mathsf{b}}(T(N), k, T(M))$ $\quad -T(\varphi_i^k N) = U_i^k(T(N))$
The set of reduction rules of the substitution calculus Red_S contains all the rules above except rule σ-generation. Recall the orientation $\succ$ from left to right of the equations in $\mathsf{Eq}_{S_{\mathsf{b}}}$. We have that $T(M) \succ T(N)$ iff $M \longrightarrow N \in \mathsf{Red}_S$. By simple calculations one can check that indeed T satisfies the conditions (BET), (GS1) and (GS2). □

As consequence of the lemma above, Theorem 3.3 and Theorem 3.5 we obtain the following two results.

Corollary 4.2
1. The substitution subcalculus of λs is confluent.
2. The calculus λs is confluent.

In a similar way, we one can show that both λt and $\lambda \upsilon$ implements the good notion of substitution S_{t}. We give only the proof for $\lambda \upsilon$.

Beta	$(\lambda M)N \to M[N/]$
App	$(MN)[s] \to (M[s])(N[s])$
Lambda	$(\lambda M)[s] \to \lambda(M[\Uparrow(s)])$
FVar	$V(1)[M/] \to M$
RVar	$V(S(n))[M/] \to V(n)$
FVarShift	$V(1)[\Uparrow(s)] \to V(1)$
RVarShift	$V(S(n))[\Uparrow(s)] \to V(n)[s][\uparrow]$
VarShift	$V(n)[\uparrow] \to V(S(n))$

Fig. 2. The rewriting system $\lambda\upsilon$

5 $\lambda\upsilon$ implements a good notion of substitution

The following definitions for the terms and rules of the $\lambda\upsilon$-calculus are taken from [14]. The terms of the $\lambda\upsilon$ are given by the grammar:

$$\begin{array}{lll} \text{Terms} & M & ::= \quad V(n) \mid MN \mid \lambda M \mid M[s] \\ \text{Substitutions} & s & ::= \quad M/ \mid \Uparrow(s) \mid \uparrow \\ \text{Naturals} & n & ::= \quad S(n) \mid 1 \end{array}$$

Note that $V(n)$ represents the variable corresponding to the number n and $S(n)$ is the successor of n. Here is an example of a reduction sequence in $\lambda\upsilon$:
$(\lambda\lambda 124)(12) \to (\lambda 124)[(12)/] \to \lambda((124)[\Uparrow(12/)]) \to \lambda 1((1[12/][\uparrow])(3[12/][\uparrow])) \to$
$\lambda 1((12)[\uparrow])(2[\uparrow]) \to 1(23)3$

We denote by $u_0^{(k)}(N)$ the term $u_0(u_0(\ldots(N)\ldots))$ where u_0 is applied k times.

Lemma 5.1 *$\lambda\upsilon$ implements the good notion of substitution S_{t}.*

Proof. We build the interpretation function T as below. The ordering $\succ$ of the set $\mathsf{Eq}_{S_{\mathsf{t}}}$ is the one given by Definition 2.2, where the equalities are oriented from left to right.

$$-T(n) = n \qquad -T(M[\Uparrow^k(\uparrow)]) = u_k(T(M))$$

$$-T(MN) = T(M)T(N) \qquad -T(M[\Uparrow^k(N/)]) = S_{\mathsf{t}}(u_0^{(k)}(T(N)), k+1, T(M))$$

$$-T(\lambda M) = \lambda T(M)$$

The proof that (BET) holds is direct using the definition of T. (GS1) holds as well. We give only the proof for (GS2). We take only the equations that perform substitution and updating on terms being variables, the cases for

terms being abstractions and application are direct by the definition of the rules of $\lambda\upsilon$.

- Let $S_{\mathsf{t}}(u_0^{(k-1)}(T(N)), k, m) \succ m-1$ with $m > k$, m, k positive natural numbers. Then, from the definition of the translation T, we have $S_{\mathsf{t}}(u_0^{(k-1)}(T(N)), k, m) = T(m[\Uparrow^{k-1}(N/)])$. We have the valid sequence of substitution reductions
  $m[\Uparrow^{k-1}(N/)] \xrightarrow{k-1} (m-k+1)[N/][\uparrow]^{k-1} \to (m-k)[\uparrow]^{k-1} \xrightarrow{k-1} m-1$.
  We conclude knowing that $T(m-1) = m-1$.
- Let $S_{\mathsf{t}}(u_0^{(k-1)}(T(N)), k, m) \succ m$ with $m < k$, m, k positive natural numbers. Then, $S_{\mathsf{t}}(u_0^{(k-1)}(T(N)), k, m) = T(m[\Uparrow^{k-1}(N/)])$, $T(m) = m$ and $m[\Uparrow^{k-1}(N/)] \xrightarrow{m-1} 1[\Uparrow^{k-m}(N/)][\uparrow]^{m-1} \to 1[\uparrow]^{m-1} \xrightarrow{m-1} m$.
- Let $S_{\mathsf{t}}(u_0^{(k-1)}(T(N)), k, k) \succ u_0^{(k-1)}(T(N))$. Then, $S_{\mathsf{t}}(u_0^{(k-1)}(T(N)), k, k) = T(k[\Uparrow^{k-1}(N/)])$. Also, $u_0^{(k-1)}(T(N)) = T(N[\uparrow]^{k-1})$ and $k[\Uparrow^{k-1}(N/)] \xrightarrow{k-1} 1[N/][\uparrow]^{k-1} \to N[\uparrow]^{k-1}$.
- Let $u_i(m) \succ m+1$ with $m > i$, m, i positive natural numbers.
  We have $u_i(m) = T(m[\Uparrow^i(\uparrow)])$ and $m[\Uparrow^i(\uparrow)] \xrightarrow{i} (m-i)[\uparrow]^{i+1} \xrightarrow{i+1} m+1$.
- Let $u_i(m) \succ m$ with $m \leq i$, m, i positive natural numbers. We have $u_i(m) = T(m[\Uparrow^i(\uparrow)])$ and $m[\Uparrow^i(\uparrow)] \xrightarrow{m-1} 1[\Uparrow^{i-m+1}(\uparrow)][\uparrow]^{m-1} \to 1[\uparrow]^{m-1} \xrightarrow{m-1} m$.

□

We can now state the following results about $\lambda\upsilon$.

Corollary 5.2
1. The substitution subcalculus of $\lambda\upsilon$ is confluent.
2. The calculus $\lambda\upsilon$ is confluent.

6 Applying our criteria for calculi of explicit substitutions with open terms

We show that indeed our method remains valid even for calculi extended with open terms. The property (GS2) indicates exactly what rules need to be added to the explicit substitution calculus such that confluence on open terms is obtained. We can use λ-calculus as the interpretation calculus since λ-calculus extended with constants is confluent. We will simplify, using our method, the proof for λse.

The calculus λse. We extend the set of terms of λs with open terms as follows:

$$\Lambda s_{\text{op}} \ ::= \ V \mid \mathbb{N} \mid (\Lambda s_{op} \Lambda s_{op}) \mid (\lambda \Lambda s_{op}) \mid (\Lambda s_{op} \sigma^i \Lambda s_{op}) \mid (\varphi_k^j \Lambda s_{op})$$

where $i, j \geq 1, k \geq 0$ and V is a set of meta-variables, over which $X, Y, Z, \ldots$ range. We do not modify at all the rules of λs. We ask the question: Is the calculus with the set of terms Λs_{op} and the rules being exactly those of λs implementing S_{b}? It is obvious that (BET) and (GS1) hold since the only change we made to the syntax of λs was the set of meta-variables. The condition (GS2) does not hold. Take the example:

$$((\lambda X)Y)\sigma^1 1 \longrightarrow (X\sigma^1 Y)\sigma^1 1 \text{ and } ((\lambda X)Y)\sigma^1 1 \longrightarrow ((\lambda X)\sigma^1 1)(Y\sigma^1 1)$$

but the terms $(X\sigma^1 Y)\sigma^1 1$ and $((\lambda X)\sigma^1 1)(Y\sigma^1 1)$ have no common reduct in λs. Indeed, $((\lambda X)\sigma^1 1)(Y\sigma^1 1) \longrightarrow (\lambda(X\sigma^2 1))(Y\sigma^1 1) \longrightarrow (X\sigma^2 1)\sigma^1(Y\sigma^1 1)$. But $T((X\sigma^1 Y)\sigma^1 1) \succ T((X\sigma^2 1)\sigma^1(Y\sigma^1 1))$ because of the Substitution Lemma for S_{b}. What is happening with the introduction of meta-variables is the "freezing" of the operation of substitution until all the meta-variables are substituted with terms containing no meta-variables. This is remarked in [12] too: "the solution of the problem seems at hand if one has in mind the properties of meta-substitutions and updating functions of the λ-calculus with de Bruijn notation." A similar observation is made also in [10, p. 7]. This is what causes the failure of (GS2) for λs with open terms.

Lemma 6.1 *The calculus λs with open terms does not implement S_{b}.*

Proof. Property (GS2) fails as explained above. □

Because of the presence of meta-variables, the set $\mathsf{Eq}_{S_{\mathsf{b}},\succ}$ is extended with oriented equations corresponding to meta-properties of S_{b} as explained at the end of Section 2. To restore (GS2) one must add to the set of rules of λse those corresponding to the oriented equations representing the meta-properties of S_{b}. The new obtained calculus λse has now in addition to the rules of λs the rules from the table below. For our example above, one should add the rule $(M\sigma^i N)\sigma^j Q \to (M\sigma^{j+1}Q)\sigma^j(N\sigma^{j-i+1}Q)$, where $i \leq j$. In the same way, all the other necessary rules are added. The interpretation function remains defined in the same way as for λs. It is easy to show that λse implements the good notion of substitution S_{b}.

Lemma 6.2 *λse implements the good notion of substitution S_{b}.*

Proof. The properties (BET) and (GS1) hold for λse as they hold for λs. The property (GS2) is restored by the addition of the new rules. □

Corollary 6.3

1. *The substitution subcalculus of λse is confluent.*
2. *The calculus λse is confluent.*

σ-σ-transition	$(M\sigma^i N)\sigma^j Q$	$\to (M\sigma^{j+1}Q)\sigma^j(N\sigma^{j-i+1}Q)$	$i \leq j$
σ-φ-transition 1	$(\varphi_k^i M)\sigma^j N$	$\to \varphi_k^{i-1} M$	$k < j < k+i$
σ-φ-transition 2	$(\varphi_k^i M)\sigma^j N$	$\to \varphi_k^i(M\sigma^{j-i+1}N)$	$k+i \leq j$
φ-σ-transition	$\varphi_k^i(M\sigma^j N)$	$\to (\varphi_{k+1}^i M)\sigma^j(\varphi_{k+1-j}^i N)$	$j \leq k+1$
φ-φ-transition 1	$\varphi_k^i(\varphi_l^j M)$	$\to \varphi_l^j(\varphi_{k+1-j}^i M)$	$l+j \leq k$
φ-φ-transition 2	$\varphi_k^i(\varphi_l^j M)$	$\to \varphi_l^{j+i-1} M$	$l \leq k < l+j$

Fig. 3. The new rules of the λse calculus.

Proof. Use Lemma 6.2, Theorem 3.3 and Theorem 3.5. □

Similar ideas for restoring (GS2) is applied for other calculi as well, e.g., $\lambda\sigma$ to obtain $\lambda\sigma_{\Uparrow}$. A different idea is used to obtain $\lambda\zeta$ from $\lambda\upsilon$, where the reduction of a β-redex is performed before the propagation of substitutions via two kinds of application operators that both correspond to the application operator in the λ-calculus.

Comparing the two methods for showing confluence of λse. We present first the method of [12]. The proof for confluence of the substitution subcalculus has two steps: weak normalization of the substitution subcalculus and a step similar to our property (GS1), see Theorem 10 and Proposition 1 of the same source. The proof of the confluence of λse is based as well on GIM. The steps of the proof are:
1. Characterize the subset, NF, of normal forms of Λs_{op} (Theorem 8). Note that $\mathsf{NF} \neq \Lambda_{\text{DB}}$.
2. Show that the substitution subcalculus is weakly normalizing: for each term $M \in \Lambda s_{\text{op}}$, $s_e^*(M)$ is a normal form of M (Theorem 10), where $s_e^* : \Lambda s_{\text{op}} \longrightarrow \mathsf{NF}$.
3. Prove a similar property to (GS1) but take $T = s_e^*$ (Proposition 1).
4. Use 2. and 3. to show confluence of the substitution subcalculus (Theorem 11).
5. Define the β-reduction , $\longrightarrow_{\text{Beta}'}$ on terms in NF such that:

$$\forall M, N \in \mathsf{NF}. M \longrightarrow_{\text{Beta}'} N \text{ iff } M \longrightarrow_{\text{Beta}} N' \text{ such that } s_e^*(N') = N$$

where $N' \in \Lambda s_{\text{op}}$. Remark the similarity of this condition with (BET) of our method.
5. Show that $\longrightarrow_{\text{Beta}'}$ is confluent using the standard method of parallel reduction.
6. Show confluence of λse using GIM, where R' is $\longrightarrow_{\text{Beta}'}$.

Our method simplifies the proofs for λse in the following ways:
1. The middle level defined in the previous proof is not necessary in our proof. So the Steps 5 and 6 from above are not required.
2. The conditions (BET) and (GS1) are checked in both methods. In the first method, where (GS1) is checked at the middle level, the proofs are more involved (see Steps 1,2,3). We check (GS1) for an interpretation function to the λ-calculus. Step 2 is easily proved from (GS2).
3. In order to apply GIM method and obtain confluence, we must check in addition to (BET) and (GS1), the condition (GS2). This checking is simpler and shorter than the Step 6 above.

7 Applying our method for calculi with composition of substitution

In the calculi of explicit substitution with composition, terms and substitutions belong to two distinct sorts. A first step is, therefore, to extend the function T from our criteria to translate substitutions (or functions or environments) to corresponding substitutions in λ-calculus. This is possible to do: $T(\sigma) = \sigma'$ such that $\forall i \quad \sigma'(i) = T(\sigma(i))$, where σ is a substitution. We also need a notion of simultaneous substitution in the sense of [17], where Stoughton gives a notion of simultaneous substitution for the λ-calculus. The author works with variable names, but one can easily take his approach to de Bruijn indices. We define the simultaneous substitution σ applied to a term M, notation $M\sigma$, as follows:

$$\text{- } k\sigma = \sigma(k) \qquad \text{- } (MN)\sigma = (M\sigma)(N\sigma) \qquad \text{- } (\lambda M)\sigma = \lambda(M\sigma')$$

where $\sigma' = 1 \cdot (\sigma \circ \uparrow)$ and the lift substitution, denoted by $\uparrow$, is defined as: $\forall k.\ \uparrow(k) = k+1$, and the cons substitution is defined as: $(M \cdot \sigma)(1) = M$ and $\forall k > 1.\ (M \cdot \sigma)(k) = \sigma(k-1)$. The composition of two substitution is defined as follows: $(\sigma \circ \tau)(k) = \sigma(k)\tau$. We define β-reduction with simultaneous substitution as the compatible closure of

$$(\lambda M)N \longrightarrow_\beta M\sigma \text{ where } \sigma = N \cdot \iota,$$

where ι is the identity substitution. β-reduction is extended to substitutions such that $\sigma_1 \longrightarrow_\beta \sigma_2$ iff $\exists i$ such that $\sigma_1(i) \longrightarrow_\beta \sigma_2(i)$ and $\forall j \neq i$ we have $\sigma_1(j) = \sigma_2(j)$. The simultaneous substitution as defined above is a good substitution. We have properties like closure of β-reduction under the simultaneous substitution, i.e., if $M \longrightarrow_\beta M'$ and $\sigma_1 \longrightarrow_\beta \sigma_2$ then $M\sigma_1 \xrightarrow{*}_\beta M'\sigma_2$. We can prove also that β-reduction defined as above is confluent. Some meta-properties of the simultaneous substitution are given below.

$(\sigma_1 \circ \sigma_2) \circ \sigma_3 = \sigma_1 \circ (\sigma_2 \circ \sigma_3) \quad (M \cdot \sigma) \circ \tau = M\tau \cdot (\sigma \circ \tau) \quad id \circ \sigma = \sigma.$
We believe that calculi in $\lambda\sigma$-style, ΛCCL [6], the suspension calculus [16] can all be shown to implement such a simultaneous substitution. The advantages of this method is that calculi in $\lambda\sigma$-style (see [14] for an overview of such calculi) can be interpreted into the common calculus of λ-calculus with a notion of simultaneous substitution as defined above, which is known to be confluent even extended with constants. However, we cannot reason about the extension of these calculi with meta-variables ranging over substitutions.

8 Conclusion and future work

In this paper, we gave a general method for the proof of confluence (ground and on open terms) for calculi of explicit substitution like λs and $\lambda \upsilon$. The key elements of our method are: the properties of a good substitution, in special preservation of β-reduction, and the existence of an interpretation from terms of the explicit substitution calculus to terms of the pure λ-calculus, satisfying three simple properties.

We plan to extend our study to $\lambda\zeta$ and also to $\lambda\sigma$. We should also consider the weak calculi of explicit substitution, e.g., $\lambda\sigma_{\mathrm{cw}}$ or $\lambda\sigma_{\mathrm{w}}$[9]. In weak calculi of explicit substitution calculi, β-reduction is forbidden under λ. We believe that our method can be extended in a natural way to other calculi, and so offers a simple proof method for the confluence of calculi of explicit substitution. We have briefly explained above how to apply our method for calculi with composition of substitutions. Another point we have to mention is that our method covers only calculi that are at least weakly-terminating. It can be argued that this is necessary since one of the main purposes of explicit substitution calculi has been to perform the process of substitution in a step-by-step manner. It is reasonable to require that there must be at least one strategy that completes such process.

Acknowledgments. I wish to thank the referees, F. Kamareddine and J. B. Wells for their helpful remarks.

References

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

[2] Z.-El-Abidine Benaissa, D. Briaud, P. Lescanne, and J. Rouyer-Degli. $\lambda\upsilon$, a calculus of explicit substitutions which preserves strong normalisation. *Journal of Functional Programming*, 6(5):699–722, 1996.

[3] P.-L. Curien and T. Hardin and J.-J. Lévy. Confluence properties of weak and strong calculi of explicit substitutions. *Journal of the ACM"*, 43(2):342–397, 1999.

[4] H. B. Curry and R. Feys. *Combinatory Logic*, volume 1. North-Holland, 1958.

[5] N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3:69–116, 1987.

[6] J. Fields. On Laziness and Optimality in Lambda Interpreters: Tools for Specification and Analysis. *Seventeenth Annual ACM Symposium on Principles of Programming Languages*, pp. 1–15, ACM press, 1990.

[7] R. David, B. Guillaume. A lambda-calculus with explicit weakening and explicit substitution. Math. Struct. in Comp. Science, 11:169–206, 2001.

[8] 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):381–392, 1972.

[9] T. Hardin. Confluence Results for the Pure Strong Categorical Logic CCL: λ-calculi as Subsystems of CCL. *Theoretical Computer Science*, 65(2):291–342, 1989.

[10] C. A. Muñoz Hurtado. Confluence and preservation of strong normalization in an explicit substitutions calculus. *Rapport de recherche, 2762*, INRIA, December, 1995.

[11] F. Kamareddine and A. Ríos. A λ-calculus à la de Bruijn with explicit substitution. In *7th Int'l Symp. Prog. Lang.: Implem., Logics & Programs, PLILP '95*, volume 982 of *LNCS*, pages 45–62. Springer-Verlag, 1995.

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

[13] F. Kamareddine and A. Ríos. Relating the $\lambda\sigma$- and λs-Styles of Explicit Substitutions. *Journal of Logic and Computation*, vol. 10, nr. 3,pp. 399-431, 2000.

[14] 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*, pp. 60–69, Jan. 1994.

[15] G. Nadathur, D. S. Wilson. A Representation of Lambda Terms Suitable for Operations on their Intensions. *Proceedings of the 1990 ACM Conference on Lisp and Functional Programming*, pp. 341–348, ACM Press, 1990.

[16] G. Nadathur, D. S. Wilson. A Notation for Lambda Terms: A Generalization of Environments. *Theoretical Computer Science*, 198(1-2): 49–98, 1998.

[17] A. Stoughton. Substitution revisited. *Theoretical Computer Science*, 59:317–325, 1988.