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 $\lambda$-calculi of explicit substitutions that admit open expressions, i.e. expressions with meta-variables. In particular, we propose a variant of the $\lambda$-calculus that we call $\lambda_C$. For this calculus and its simply-typed version, we study its meta-theoretical properties. The $\lambda_C$-calculus enjoys the same general characteristics as $\lambda$, i.e. a simple and finitary first-order presentation, confluent on expressions with meta-variables of terms and weakly normalizing on typed expressions. Moreover, $\lambda_C$ 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 $\lambda$-calculi of explicit substitutions (see, among others, [1, 20, 14, 2, 21, 3, 17, 25, 6]). All these calculi implement $\beta$-reductions by means of a lazy mechanism of reduction of substitutions.

In typed $\lambda$-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 $\lambda$-calculi of explicit substitutions with meta-variables.

The $\lambda\sigma$-calculus\footnote{In this paper we use $\lambda\sigma$ to designate the locally confluent calculus proposed in [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, $\lambda$-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. $\lambda_T$ [3], $\lambda_{S_e}$ [14] or $\lambda_\zeta$ [25]), the $\lambda\sigma$-calculus is a finitary first-order system ($\lambda_{S_e}$ is not),
it allows composition and simultaneous substitutions ($\lambda_\Box$ does not), and it is compatible with the extensional $\eta$-rule ($\lambda_\Pi$ 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 $\lambda$-calculus can be found in [29].

However, composition of substitutions and simultaneous substitutions are responsible of the following non-left-linear rule in $\lambda_\sigma$: $1[S] \cdot (\uparrow o S) \xrightarrow{(\text{SCons})} S$. Informally, if we interpret $S$ as a list, $1$ 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: $1[\uparrow^n] \cdot \uparrow^{n+1} \xrightarrow{(\text{SCons})} \uparrow^n$, where $\uparrow^n$ is a notation for $\uparrow \circ \ldots \circ \uparrow$.

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 $\lambda_\Box$-calculus. The confluence property of $\lambda_\Box$ is show in Section 3. In Section 4 we study the simply-typed version of $\lambda_\Box$. In Section 5 we prove that $\lambda_\Box$ is weakly normalizing on typed expressions. Last section summarizes the main contributions of this work.

2 $\lambda_\Box$-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 $\lambda_\Box$ are defined by the following grammar:\textsuperscript{2}

\begin{align*}
\text{Naturals} & \quad n \quad ::= \quad 0 \mid \text{Suc}(n) \\
\text{Terms} & \quad M, N \quad ::= \quad 1 \mid \lambda M \mid (M \cdot N) \mid M[S] \\
\text{Substitutions} & \quad S, T \quad ::= \quad \uparrow^n \mid M \cdot S \mid S \circ T
\end{align*}

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