The $\lambda\sigma$-Calculus and Strong Normalization

Anders Schack-Nielsen
Carsten Schürmann

IT University Technical Report Series **TR-2011-TR-2011-150**

ISSN 1600–6100 **December 2011**

ISSN 1600–6100

ISBN 978-87-7949-249-3

Copies may be obtained by contacting:

IT University of Copenhagen
Rued Langgaards Vej 7
DK-2300 Copenhagen S
Denmark

Telephone: +45 72 18 50 00
Telefax: +45 72 18 50 01
Web `www.itu.dk`

The $\lambda\sigma$-Calculus and Strong Normalization

Anders Schack-Nielsen Carsten Schürmann
IT University of Copenhagen

No Institute Given

Abstract. Explicit substitution calculi can be classified into several distinct categories depending on whether they are *confluent*, *meta-confluent*, *strong normalization preserving*, *strongly normalizing*, *simulating*, *fully compositional*, and/or *local*. In this paper we present a variant of the $\lambda\sigma$-calculus, which satisfies all seven conditions. In particular, we show how to circumvent Mellies counter-example to strong normalization by a slight restriction of the congruence rules. The calculus is implemented as the core data structure of the Celf logical framework. All meta-theoretic aspects of this work have been mechanized in the Abella proof assistant.

1 Introduction

Explicit substitution calculi are useful when implementing λ-calculi with meta-variables, and serve therefore as the formal foundation for many proof assistants. By making substitutions explicit, however, some properties, such as, for example, strong normalization [Mel95] or meta-confluence [CHL96] may be broken. Different flavors of explicit substitution calculi have been studied and classified extensively, for example, by Kesner [Kes07]. One property that we refer to as *locality*, however, is absent from the usual classifications: A calculus is local, if no reduction rule relies on side conditions that inspect arbitrarily large terms. So far, all attempts to render the original $\lambda\sigma$-calculus strongly normalizing have resulted in non-local reduction rules, the loss of confluence in the presence of meta-variables, or have other undesirable side effects. In this paper we solve this problem and give a strongly normalizing version of the $\lambda\sigma$-calculus, which is elegant, concise, and satisfies the usually desired properties plus locality.

Consider the λ-calculus and its reduction semantics $\to_\beta$:

$$t ::= x \mid t_1\ t_2 \mid \lambda x.\, t$$

$$(\lambda x.\, t_1)\ t_2 \to_\beta t_1\{t_2/x\}$$

Explicit substitution calculi are usually formulated and studied without reference to meta-variables as they can always be considered an extension. However, meta-variables illustrate our interest in explicit substitution calculi well. In a λ-calculus with meta-variables, substitutions can no longer be pushed into the term until they disappear, instead they get stuck until the meta-variable is instantiated. Explicit substitutions have become popular among programming language, theorem prover, and proof assistant developers.

The simplest way to internalize substitutions is to turn $t_1\{t_2/x\}$ into a term constructor $t_1[t_2/x]$:

$$t ::= \cdots \mid t_1[t_2/x]$$

and the definition of $t_1\{t_2/x\}$ into reduction rules:

$$\begin{array}{ll} x[t/x] \to t & (t_1\ t_2)[t_3/x] \to t_1[t_3/x]\ t_2[t_3/x] \\ y[t/x] \to y \quad \text{if } x \neq y & (\lambda y.\, t_1)[t_2/x] \to \lambda y.\, t_1[t_2/x] \quad \text{if } x \neq y \text{ and } y \notin \mathrm{fv}(t_2) \end{array}$$

along with the β-rule $(\lambda x.\, t_1)\ t_2 \to t_1[t_2/x]$. This explicit substitution calculus is known as λx [Kes07,BR95]. However, without any way to compose substitutions, this calculus is not confluent in the presence of meta-variables, and a naive addition of composition rules breaks normalization.

Many different explicit substitution calculi have been proposed trying to capture as many good properties as possible. Kesner gives a nice overview of related work in [Kes07] and highlights six important and desirable properties: confluence **(C)** of the reduction relation, meta-confluence **(MC)**, i.e. confluence in the presence of meta-variables, the preservation of strong normalization **(PSN)** (which means that any explicit substitution free term that is strongly normalizing with respect to $\to_\beta$ is also strongly normalizing with respect to $\to$), strong normalization **(SN)** of well-typed terms, simulation **(SIM)** (which means the reduction relation $\to$ can simulate

beta	$(\lambda M)\ N$	$\to M[N\,.\,\mathsf{id}]$
clos-var	$1[M\,.\,s]$	$\to M$
clos-clos	$M[s][t]$	$\to M[s \circ t]$
clos-lam	$(\lambda M)[s]$	$\to \lambda(M[1\,.\,(s \circ \uparrow)])$
clos-app	$(M\ N)[s]$	$\to M[s]\ N[s]$
clos-var-id	$1[\mathsf{id}]$	$\to 1$
comp-id-L	$\mathsf{id} \circ s$	$\to s$
comp-shift-id	$\uparrow \circ \mathsf{id}$	$\to \uparrow$
comp-shift	$\uparrow \circ (M\,.\,s)$	$\to s$
comp-cons	$(M\,.\,s) \circ t$	$\to M[t]\,.\,(s \circ t)$
comp-comp	$(s_1 \circ s_2) \circ s_3$	$\to s_1 \circ (s_2 \circ s_3)$

Fig. 1. Basic reduction rules

β-reduction, i.e. $t_1 \to_\beta t_2$ implies $t_1 \to^* t_2$), and finally, full composition **(FC)** (which states that for any t_1 and t_2 it holds that $t_1[t_2/x] \to^* t_1\{t_2/x\}$ for an appropriate extension of ordinary capture-avoiding substitution to terms with explicit substitutions). In addition, we propose to consider the property we refer to as locality **(L)**, where we define calculus to be local, if no reduction rule relies on side conditions that inspect arbitrarily large terms.

All the rules of λx above are local. However, the rule $\mathbf{gc}$ below that is known from several explicit substitution calculi [Kes07,BR95] is not.

$$(\mathbf{gc}) \qquad t_1[t_2/x] \to t_1 \quad \text{if } x \notin \mathrm{fv}(t_1)$$

This side condition differs from checking for α-equivalence, because the structure of t_1 must be traversed, and this operation can hence not be executed in constant time.

The $\lambda\sigma$-calculus is a simple minimalistic explicit substitution calculus, which has most of the properties we desire from an explicit substitution calculus (**C**, **SIM**, **FC**, **L**) [Kes09].

In this paper we show how a small restriction in the reduction rules allows us to recover the remaining three properties **PSN**, **SN**, and **MC**. The meta-theory of this paper has been completely mechanized in the proof assistant Abella, and each theorem is marked with the corresponding Abella file and theorem name. The Abella source files can be found at `http://www.itu.dk/people/anderssn/exsub-sn.tgz`. The proofs have been incorporated into the Abella example collection where they can be easily browsed: `http://abella.cs.umn.edu/examples/lambda-calculus/exsub-sn/`.

2 Preliminaries

The syntax of the $\lambda\sigma$-calculus consists of terms and substitutions written in de Bruijn notation:

Terms:	$M, N ::= 1 \mid M[s] \mid \lambda M \mid M\ N$
Substitutions:	$s, t ::= \mathsf{id} \mid \uparrow \mid M\,.\,s \mid s \circ t$

The variable 1 refers to the innermost λ-binder. Other variables are represented with a closure and a sequence of shifts, e.g. $3 = 1[\uparrow \circ \uparrow]$.

The intuition behind each of the substitution constructs is the following: A term under an identity (id) is supposed to reduce to itself. A shift ($\uparrow$) applied to a term increments all freely occurring variables by one. An extension $M\,.\,s$ will substitute M for the variable 1, decrement all other freely occurring variables, and then apply s to them. Finally, a composition of two substitutions $s \circ t$ represents the substitution that first applies s and then t, i.e. $M[s \circ t]$ is supposed to reduce to the same term as reducing each closure individually in $M[s][t]$.

We will use $\uparrow^n$ where $n \geq 0$ as a short-hand for n compositions of shift, i.e. $\uparrow \circ (\uparrow \circ (\ldots \circ (\uparrow \circ \uparrow) \ldots))$, where $\uparrow^0$ means id. Additionally, de Bruijn indices n with $n > 1$ are short-hand for $1[\uparrow^{n-1}]$.

The reduction rules are shown in Figure 1. In addition to these rules we consider every possible congruence rule, i.e. we allow rewrites to occur anywhere inside a term or substitution.

The rule **beta** corresponds to the ordinary β-step in the λ-calculus and introduces an explicit substitution inside a closure. The rest of the rules are called σ-rules and details how to evaluate closures and substitution

beta	$(\lambda M)\ N$	$\to M[N\ .\ \mathsf{id}]$
clos-const	$c[s]$	$\to c$
clos-var-dot1	$1[M\ .\ s]$	$\to M$
clos-var-dot2	$(n+1)[M\ .\ s]$	$\to n[s]$
clos-var-shift	$n[\uparrow^m]$	$\to n+m$
clos-clos	$M[s][t]$	$\to M[s \circ t]$
clos-lam	$(\lambda M)[s]$	$\to \lambda(M[1\ .\ (s \circ \uparrow^1)])$
clos-app	$(M\ N)[s]$	$\to M[s]\ N[s]$
comp-id-L	$\uparrow^0 \circ s$	$\to s$
comp-cons	$(M\ .\ s) \circ t$	$\to M[t]\ .\ (s \circ t)$
comp-shift-dot	$\uparrow^{n+1} \circ (M\ .\ s)$	$\to \uparrow^n \circ s$
comp-shift-shift	$\uparrow^n \circ \uparrow^m$	$\to \uparrow^{n+m}$
comp-comp	$(s_1 \circ s_2) \circ s_3$	$\to s_1 \circ (s_2 \circ s_3)$
eta-subst	$(n+1)\ .\ \uparrow^{n+1}$	$\to \uparrow^n$

Fig. 2. Reduction rules

compositions. The fragment of the rules that excludes the **beta** rule is called the σ-fragment and the corresponding relation is written $\to_\sigma$.

In the presence of meta-variables this calculus is not even locally confluent. The critical pair $((\lambda M)N)[s]$ can reduce to both $M[N[s]\ .\ s]$ and $M[N[s]\ .\ (s \circ \mathsf{id})]$. Closing the critical pairs gives rise to the following four additional rules:

comp-id-R	$s \circ \mathsf{id}$	$\to s$
clos-id	$M[\mathsf{id}]$	$\to M$
var-shift	$1\ .\ \uparrow$	$\to \mathsf{id}$
s-cons	$1[s]\ .\ (\uparrow \circ s)$	$\to s$

The first two rules are the general right-identity rules and thus replace **clos-var-id** and **comp-shift-id**. The **var-shift** and **s-cons** rules can be seen as η-contraction rules on substitutions. The resulting system is easily seen to be locally confluent by checking all critical pairs. Constants can be added to the calculus with a single additional rule:

$$\textbf{clos-const}\ c[s] \to c$$

We can also represent de Bruijn indices and sequences of shifts directly in the syntax with the following five rules:

clos-var-dot2	$(n+1)[M\ .\ s]$	$\to n[s]$
clos-var-shift	$n[\uparrow^m]$	$\to n+m$
comp-shift-dot	$\uparrow^{n+1} \circ (M\ .\ s)$	$\to \uparrow^n \circ s$
comp-shift-shift	$\uparrow^n \circ \uparrow^m$	$\to \uparrow^{n+m}$
eta-subst	$(n+1)\ .\ \uparrow^{n+1}$	$\to \uparrow^n$

These rules are essentially shortcuts in the sense that the first four can be obtained by a sequence of the reduction steps shown in Figure 1 when n and $\uparrow^n$ are syntactic short-hands. It is also easy to check that the addition of the first four rules suffices to evaluate all compositions and closures. These rules render **clos-var-id**, **comp-shift-id**, and **comp-shift** obsolete. The last rule generalizes **var-shift** and corresponds to **s-cons** in the case when $s = \uparrow^n$.

In summary, we base this paper on the following definition of $\lambda\sigma$:

Terms:	$M, N ::= n \mid c \mid M[s] \mid \lambda M \mid M\ N$
Substitutions:	$s, t ::= \uparrow^n \mid M\ .\ s \mid s \circ t$

where de Bruijn indices n have $n \geq 1$ and shifts $\uparrow^n$ have $n \geq 0$. The reduction rules are given in Figure 2. We have excluded the two general right-identity rules, as we can easily prove them admissible for the transitive closure:

Theorem 1 (`beta.thm:clos_id_ext`). *For all M and s we have $M[\uparrow^0] \to^* M$ and $s \circ \uparrow^0 \to^* s$.*

We have also omitted the general **s-cons** rule and only included **eta-subst** in its place. The general **s-cons** rule is problematic in the sense that it is not left-linear, so it is worth analyzing what it actually adds. It is easy to see that

any composition can reduce to either $M \,.\, s$ or $\uparrow^n$ (a concrete reduction sequence is given by Theorem 15 below). In these two cases **s-cons** gives us the two reductions $1[M \,.\, s] \,.\, (\uparrow^1 \circ (M \,.\, s)) \to M \,.\, s$ and $1[\uparrow^n] \,.\, (\uparrow^1 \circ \uparrow^n) \to \uparrow^n$. The former can be achieved in two steps without **s-cons**, and the latter corresponds to **eta-subst**. This shows that we do not lose any reduction sequences by excluding the problematic **s-cons** in favor of **eta-subst**, and thus it is the better choice.

We are not going to say much about the congruence rules yet, so for now we will just assume that we can perform any reduction step anywhere within a term or substitution. We will return to this matter below in Section 3.

Mellies showed in [Mel95] that the simply typed term

$$\lambda v.(\lambda x.(\lambda y.y)((\lambda z.z)x))((\lambda w.w)v)$$

has an infinite reduction sequence in $\lambda\sigma$. This is very counter-intuitive since the ordinary simply typed λ-calculus is strongly normalizing and the σ-fragment of $\lambda\sigma$ is also strongly normalizing [CHR92] (even on untyped terms).

We sketch the idea of the counter-example. Consider a β-redex under a closure $((\lambda M)\ N)[s]$ If we evaluate the redex first and then the substitution composition we arrive at $M[N[s].s]$. If we instead begin by pushing the substitution through the application we can get the following reduction sequence:

$$\begin{aligned}((\lambda M)\ N)[s] &\to (\lambda M)[s]\ N[s] \\ &\to (\lambda M[1 \,.\, s \circ \uparrow^1])\ N[s] \\ &\to M[1 \,.\, s \circ \uparrow^1][N[s] \,.\, \uparrow^0] \\ &\to^* M[N[s] \,.\, s \circ (\uparrow^1 \circ (N[s] \,.\, \uparrow^0))]\end{aligned}$$

Here we see a substitution $s' = \uparrow^1 \circ (N[s] \,.\, \uparrow^0)$, which contains s, being applied to s itself. Of course s' can reduce in one step to $\uparrow^0$, but if we carefully avoid that specific reduction step and if s contains a β-redex then we can push s' into s and through the redex in s and thus replicate the situation above with s' instead of s. Now since s' contains s and therefore also a β-redex we can keep on doing this (see [Mel95] for all the details).

The term that we end up creating in this way consists of a sequence of closures nested arbitrarily deep:

$$M[\ldots M[\ldots M[\ldots M[\ldots]\ldots]\ldots]\ldots]$$

And consequently the reduction steps that we perform are similarly nested deeper and deeper through an arbitrary number of closures and substitutions.

This also highlights the brittle nature of the counter-example. If we at any time were to reduce any of the arising s's to $\uparrow^0$ the entire thing would normalize. Thus we have an indication that a suitable minor tweak to the reduction strategy will give us strong normalization (as opposed to the current non-deterministic, everything-is-allowed reduction strategy).

3 Regaining Strong Normalization

Let us present the congruence rules that up until now have remained implicit:

$$\frac{M \to M'}{M \,.\, s \to M' \,.\, s} \qquad \frac{s \to s'}{M \,.\, s \to M \,.\, s'} \qquad \frac{s \to s'}{s \circ t \to s' \circ t} \qquad \frac{t \to t'}{s \circ t \to s \circ t'}$$

$$\frac{M \to M'}{\lambda M \to \lambda M'} \qquad \frac{M \to M'}{M\ N \to M'\ N} \qquad \frac{N \to N'}{M\ N \to M\ N'}$$

$$\frac{M \to M'}{M[s] \to M'[s]} \qquad \frac{s \to s'}{M[s] \to M[s']}\ (*)$$

In order to prevent Mellies' counter example, we will have to disallow certain reductions within a term. If we get rid of the second congruence rule for closures $(*)$ then certainly we will have a strongly normalizing calculus, but this is overly restrictive. If we instead replace it by a version that only allows σ-steps then since the σ-fragment by itself is strongly normalizing, we can hope for strong normalization. We are therefore going to use the following congruence rule in place of $(*)$:

$$\frac{s \to_\sigma s'}{M[s] \to_\sigma M[s']}$$

$$
\begin{array}{llll}
c[e] & \to c & 1[e_1 \,.\, e_2] & \to e_1 \\
(n+1)[e_1 \,.\, e_2] & \to n[e_2] & n[\uparrow^m] & \to n+m \\
e_1[e_2][e_3] & \to e_1[e_2[e_3]] & (\lambda e_1)[e_2] & \to \lambda e_1[1 \,.\, e_2[\uparrow^1]] \\
(e_1 \,.\, e_2)[e_3] & \to e_1[e_3] \,.\, e_2[e_3] & \uparrow^0[e] & \to e \\
\uparrow^{n+1}[e_1 \,.\, e_2] & \to \uparrow^n[e_2] & \uparrow^n[\uparrow^m] & \to \uparrow^{n+m} \\
(n+1) \,.\, \uparrow^{n+1} & \to \uparrow^n & &
\end{array}
$$

$$
\frac{e_1 \to e_1'}{e_1[e_2] \to e_1'[e_2]} \qquad \frac{e_2 \to e_2'}{e_1[e_2] \to e_1[e_2']} \qquad \frac{e \to e'}{\lambda e \to \lambda e'}
$$

$$
\frac{e_1 \to e_1'}{e_1 \,.\, e_2 \to e_1' \,.\, e_2} \qquad \frac{e_2 \to e_2'}{e_1 \,.\, e_2 \to e_1 \,.\, e_2'}
$$

Fig. 3. Expression reduction rules

Now, β-steps may not be performed in closures, however the resulting rewrite system is still **C**, **SIM**, **FC**, and **L**: we can always reduce the closure itself. In addition, it is strongly normalizing for simply typed terms as we will discuss next.

Strong normalization of $\lambda\sigma$. The proof of strong normalization can be summarized to the following: Take any reduction sequence and divide it into groups of σ-steps and **beta**-steps (applications of the **beta**-rule). We relate each term to its σ-normal form and then we show **PSN** by proving that every **beta**-step in $\lambda\sigma$ corresponds to a β-reduction step in the λ-calculus.

Strong normalization of the σ-fragment. In [CHR92] the σ-fragment of $\lambda\sigma$ was shown to be strongly normalizing. Central to their argument was the proof that if e is strongly normalizing then $e[\uparrow^1]$ is also strongly normalizing, where e is an expression (defined below). The proof we give in this section shares the same overall structure, but we would like to emphasize that our proof of the strong normalization of $e[\uparrow^1]$ given strong normalization of e is greatly simplified (Lemma 4 and Lemma 5 below).

In order to give a simpler proof of strong normalization of the σ-fragment, we collapse the syntactic classes of terms and substitutions into a single class called expressions:

$$\textbf{Expressions:} \qquad e ::= n \mid c \mid \uparrow^n \mid e_1[e_2] \mid \lambda e \mid e_1 \,.\, e_2$$

The set of reduction rules for expressions is given in Figure 3. We use the following translation from terms and substitutions into expressions:

$$\mathcal{E}(n) = n \qquad \mathcal{E}(c) = c \qquad \mathcal{E}(M[s]) = \mathcal{E}(M)[\mathcal{E}(s)]$$

$$\mathcal{E}(\lambda M) = \lambda\mathcal{E}(M) \qquad \mathcal{E}(M\ N) = \mathcal{E}(M) \,.\, \mathcal{E}(N) \qquad \mathcal{E}(\uparrow^n) = \uparrow^n$$

$$\mathcal{E}(M \,.\, s) = \mathcal{E}(M) \,.\, \mathcal{E}(s) \qquad \mathcal{E}(s \circ t) = \mathcal{E}(s)[\mathcal{E}(t)]$$

It is easy to see that σ-reductions are preserved by the translation, and therefore strong normalization of expressions implies strong normalization of the σ-fragment.

We shall write $\mathcal{SN}$ for the set of strongly normalizing expressions, and then aim to prove $e \in \mathcal{SN}$ for all expressions e. Strong normalization of the cases λe and $e_1 \,.\, e_2$ are easy.

Lemma 1 (`sigma-strong.thm:esn_lam`). *If $e \in \mathcal{SN}$ then $\lambda e \in \mathcal{SN}$.*

Lemma 2 (`sigma-strong.thm:esn_dot,esn_dot_inv`). *$e_1 \,.\, e_2 \in \mathcal{SN}$ if and only if $e_1 \in \mathcal{SN}$ and $e_2 \in \mathcal{SN}$.*

Closures are a lot more difficult due to the rewrite $(\lambda e_1)[e_2] \to \lambda e_1[1 \,.\, e_2[\uparrow^1]]$. We will therefore split the proof depending on whether the given expressions contain λs. We write the exclusion of λs from an expression e as $\lambda \notin e$.[1]

[1] The formalized proof represents the exclusion of λ by a predicate `nolam`. This predicate also excludes constants to reduce the number of cases needed in the proofs, but it could just as well have included them, and the general theorem is proved later anyway. Thus, whenever we write $\lambda \notin e$ we will also exclude occurrences of constants in e.

Lemma 3 (`sigma-strong.thm:esn_clos_nolam`). *If $e_1 \in \mathcal{SN}$, $e_2 \in \mathcal{SN}$, and $\lambda \notin e_1$ then $e_1[e_2] \in \mathcal{SN}$.*

Theorem 2 (`sigma-strong.thm:nolam_esn`). *If $\lambda \notin e$ then $e \in \mathcal{SN}$.*

In order to prove the general statement we are going to need a version of Lemma 3 without the restriction on e_1. The tricky part is proving that $e_2 \in \mathcal{SN}$ implies $e_2[\uparrow^1] \in \mathcal{SN}$ (and thus $1 \,.\, e_2[\uparrow^1] \in \mathcal{SN}$).

If we do an intuitive comparison between reduction sequences for e and $e[\uparrow^1]$ it seems that any reduction in $e[\uparrow^1]$ either can be mimicked by a reduction in e or consists of pushing the $\uparrow^1$ through e. We will formalize this intuitive idea by building a relation $e \overset{\triangleleft}{\sim} e'$, such that $e \overset{\triangleleft}{\sim} e[\uparrow^1]$ for any e, and for any $e_1' \to e_2'$ with $e_1 \overset{\triangleleft}{\sim} e_1'$ then either $e_1 \overset{\triangleleft}{\sim} e_2'$ or $e_1 \to e_2$ with $e_2 \overset{\triangleleft}{\sim} e_2'$. In the former case the reduction $e_1' \to e_2'$ is intended to be one of the steps associated with pushing the $\uparrow^1$ through the structure of the expression and thus the number of such steps should be bounded by the structure of the expression. Given such a relation we would get the desired lemma as a corollary to the fact that $e \in \mathcal{SN}$ and $e \overset{\triangleleft}{\sim} e'$ implies $e' \in \mathcal{SN}$.

In order to build the relation such that it is closed under the conditions stated above, we need it to contain $e \overset{\triangleleft}{\sim} e[e']$ where e' can be $\uparrow^1$ and is closed under at least $1 \,.\, \cdot \circ \uparrow^1$ and reduction. Instead of characterizing this class of expressions we can simply take expressions without λ as we already know this class to be in $\mathcal{SN}$. The relation is defined as follows:

$$\frac{}{c \overset{\triangleleft}{\sim} c} \qquad \frac{\lambda \notin e \quad \lambda \notin e'}{e \overset{\triangleleft}{\sim} e'} \qquad \frac{e_2 \overset{\triangleleft}{\sim} e_2'}{e_1[e_2] \overset{\triangleleft}{\sim} e_1[e_2']} \qquad \frac{e_1 \overset{\triangleleft}{\sim} e_1' \quad e_2 \overset{\triangleleft}{\sim} e_2'}{e_1 \,.\, e_2 \overset{\triangleleft}{\sim} e_1' \,.\, e_2'}$$

$$\frac{e \overset{\triangleleft}{\sim} e'}{\lambda e \overset{\triangleleft}{\sim} \lambda e'} \qquad \frac{\lambda \notin e'}{e \overset{\triangleleft}{\sim} e[e']} \qquad \frac{e_1 \overset{\triangleleft}{\sim} e_1' \quad \lambda \notin e_2}{e_1[e_2] \overset{\triangleleft}{\sim} e_1'[e_2]}$$

The following theorem states the desired simulation properties.

Lemma 4 (`sigma-strong.thm:is_esn_rel_under_shift`). *If $e_0 \overset{\triangleleft}{\sim} e_0'$ then there exists a k such that for all reductions $e_0' \to e_1' \to \cdots \to e_n'$ in n steps there exists a sequence $e_0, e_1, \ldots, e_n$ such that*

1. *either $e_i \to e_{i+1}$ or $e_i = e_{i+1}$ for $0 \leq i < n$,*
2. *$e_i \overset{\triangleleft}{\sim} e_i'$ for $0 \leq i \leq n$, and*
3. *$n \geq k$ implies $e_i \to e_{i+1}$ for some i.*

It is noteworthy to consider how Lemma 4 is encoded in Abella. The statement of the lemma is similar to strong normalization in the sense that some limit k exists after which any reduction sequence will bottom out in some way; in this case, have a corresponding reduction in the $\overset{\triangleleft}{\sim}$-related expression. The Abella-encoding of $\mathcal{SN}$ is the following inductive definition:

$$e \in \mathcal{SN} \quad := \quad \forall e'.\, e \to e' \supset e' \in \mathcal{SN}$$

We can then give a similar, but slightly more complicated, inductive definition of a relation $\mathcal{SN}[e]$ (denoted `esn_rel_under_shift` in the formalization).

$$\begin{aligned} e_1' \in \mathcal{SN}[e_1] \quad := \quad & e_1 \overset{\triangleleft}{\sim} e_1' \wedge \forall e_2'.\, e_1' \to e_2' \supset \\ & (\exists e_2.\, e_1 \to e_2 \wedge e_2 \overset{\triangleleft}{\sim} e_2') \vee e_2' \in \mathcal{SN}[e_1] \end{aligned}$$

Lemma 4 can now be represented as the statement $e \overset{\triangleleft}{\sim} e' \supset e' \in \mathcal{SN}[e]$. By a nested induction on the strong normalization of e and k from Lemma 4 we get:

Lemma 5 (`sigma-strong.thm:exp_under_shift_esn`). *If $e \in \mathcal{SN}$ and $e \overset{\triangleleft}{\sim} e'$ then $e' \in \mathcal{SN}$.*

As an immediate corollary we get that $e \in \mathcal{SN}$ implies $e[\uparrow^1] \in \mathcal{SN}$. This gives us the desired version of Lemma 3 without the restriction on e_1.

Lemma 6 (`sigma-strong.thm:esn_clos`). *If $e_1 \in \mathcal{SN}$ and $e_2 \in \mathcal{SN}$ then $e_1[e_2] \in \mathcal{SN}$.*

Together Lemmas 1, 2, and 6 yield the strong normalization theorem:

Theorem 3 (`sigma-strong.thm:exp_esn`). *Every expression is strongly normalizing: $e \in \mathcal{SN}$ for all e.*

And thus, we have strong normalization of the σ-fragment of $\lambda\sigma$.

Theorem 4 (`lambda-sigma.thm:tm_sn_su, sub_sns_su`). *The reduction relation $\to_\sigma$ is strongly normalizing for terms and substitutions.*

$$\frac{\Sigma(c) = A}{\Gamma \vdash c : A} \qquad \frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B} \qquad \frac{\Gamma, A \vdash M : B}{\Gamma \vdash \lambda M : A \to B}$$

$$\frac{}{\Gamma, A \vdash 1 : A} \qquad \frac{\Gamma \vdash n : A}{\Gamma, B \vdash n+1 : A} \qquad \frac{\Gamma \vdash s : \Gamma' \quad \Gamma' \vdash M : A}{\Gamma \vdash M[s] : A} \qquad \frac{}{\Gamma \vdash \uparrow^0 : \Gamma}$$

$$\frac{\Gamma \vdash \uparrow^n : \Gamma'}{\Gamma, A \vdash \uparrow^{n+1} : \Gamma'} \qquad \frac{\Gamma \vdash M : A \quad \Gamma \vdash s : \Gamma'}{\Gamma \vdash M\,.\,s : \Gamma', A} \qquad \frac{\Gamma \vdash t : \Gamma'' \quad \Gamma'' \vdash s : \Gamma'}{\Gamma \vdash s \circ t : \Gamma'}$$

Fig. 4. Types for $\lambda\sigma$

Confluence of the σ-fragment. With strong normalization we can easily prove confluence of the σ-fragment by first proving local confluence.

Theorem 5 (`conf.thm:local_conf`). *The σ-fragment is locally confluent.*

1. *If $M_1\ {}_\sigma\!\leftarrow M \to_\sigma M_2$ then there exists an M' s.t. $M_1 \to_\sigma^* M'\ {}_\sigma^*\!\leftarrow M_2$.*
2. *If $s_1\ {}_\sigma\!\leftarrow s \to_\sigma s_2$ then there exists an s' s.t. $s_1 \to_\sigma^* s'\ {}_\sigma^*\!\leftarrow s_2$.*

Theorem 6 (`conf.thm:conf_tm,conf_sub`). *The σ-fragment is confluent.*

1. *If $M_1\ {}_\sigma^*\!\leftarrow M \to_\sigma^* M_2$ then there exists an M' s.t. $M_1 \to_\sigma^* M'\ {}_\sigma^*\!\leftarrow M_2$.*
2. *If $s_1\ {}_\sigma^*\!\leftarrow s \to_\sigma^* s_2$ then there exists an s' s.t. $s_1 \to_\sigma^* s'\ {}_\sigma^*\!\leftarrow s_2$.*

Together confluence and strong normalization give us the existence of unique σ-normal forms. We shall denote the σ-normal form of a term M or substitution s as $\sigma(M)$ and $\sigma(s)$, respectively.

Preservation of strong normalization. With the σ-fragment covered, we turn our attention to the **beta**-steps. Let $\to_\beta$ denote a **beta**-step. Then the reduction relation $\to$ is the disjoint union of $\to_\sigma$ and $\to_\beta$. The σ-normal forms correspond to the ordinary λ-calculus, so for such terms we can define ordinary β-reduction in the following way:

$$M \Rightarrow_\beta M' \quad := \quad M \to_\beta M'' \wedge M' = \sigma(M'')$$

Preservation of strong normalization (PSN) is the property that strong normalization of $\sigma(M)$ with respect to $\Rightarrow_\beta$ implies strong normalization of M with respect to $\to$. Our restriction of the congruence rules allows us to prove that **beta**-steps correspond to β-steps in the ordinary λ-calculus. Together with **C** and **SN** of the σ-fragment, we immediately get **PSN**:

Theorem 7 (`beta.thm:project_beta`). *If $M \to_\beta M'$ then $\sigma(M) \Rightarrow_\beta \sigma(M')$.*

Theorem 8 (`strong-norm.thm:psn`). *If $\sigma(M)$ is strongly normalizing with respect to $\Rightarrow_\beta$ then M is strongly normalizing with respect to $\to$.*

Strong normalization of simply typed $\lambda\sigma$. So far, we have only considered untyped terms and substitutions. So before we can talk about strong normalization of well-typed terms and substitutions, we need to introduce the type system. The typing rules are standard for $\lambda\sigma$ and given in Figure 4.

Theorem 9 (`typing.thm:of_step_ext`). *Subject reduction.*

1. *If $\Gamma \vdash M : A$ and $M \to M'$ then $\Gamma \vdash M' : A$.*
2. *If $\Gamma \vdash s : \Gamma'$ and $s \to s'$ then $\Gamma \vdash s' : \Gamma'$.*

Since well-typed σ-normal forms are exactly the simply typed λ-calculus, we have the usual proof of strong normalization using a logical relation. The Abella proof is adapted from Girard's proof of strong normalization in the example suite of Abella [Abe].

Theorem 10 (`beta-sn.thm:strong_beta`). *Let M be a σ-normal form with $\Gamma \vdash M : A$. Then M is strongly normalizing with respect to $\Rightarrow_\beta$.*

PSN (Theorem 8) guarantees strong normalization of simply typed terms:

Theorem 11 (`strong-norm.thm:strong_tm`). *If $\Gamma \vdash M : A$ then M is strongly normalizing with respect to $\rightarrow$.*

We can adapt the proof of Lemma 3 to prove strong normalization of compositions. We will not have to consider λs in this case, since for terms we can simply appeal to Theorem 11.

Lemma 7 (`strong-norm.thm:sns_clos`). *If $\Gamma \vdash t : \Gamma''$, $\Gamma'' \vdash s : \Gamma'$, and s and t are strongly normalizing then $s \circ t$ is strongly normalizing.*

Theorem 12 (`strong-norm.thm:strong_sub`). *If $\Gamma \vdash s : \Gamma'$ then s is strongly normalizing.*

4 Meta-Confluence

We consider now a formulation of $\lambda\sigma$-calculus extended by term meta-variables as known from contextual modal type theory [NPP08]. From a term rewriting perspective we want to keep the property that any closure $M[s]$ can reduce (**FC**), and from a contextual modal type theory perspective logic variables are always under a substitution. Therefore, we do not consider the new normal form $X[s]$ to be a closure, but rather a logic variable along with a substitution, which allows all congruence rules.

The key to **MC** is the shape of normal forms. As we mentioned in Section 2, it is easy to see that compositions can reduce to either $M\,.\,s$ or $\uparrow^n$, and thus **eta-subst** can replace **s-cons**. With **eta-subst** we get confluence in the presence of term meta-variables, as we have show in [SNS10]. We even mechanized the proof in Twelf, which remains valid, because it relies on σ-normalization, and for σ-normal-forms the restricted operational semantics coincides with the unrestricted. However, adding substitution meta-variables breaks confluence [CHL96]. In practice, substitution meta-variables are of lesser interest than term meta-variables, so it is tempting to simply ignore them and declare meta-confluence achieved, but we will discuss them here for completeness.

The counterexample to confluence works by creating two diverging reduction paths through the clever use of **s-cons**, which cannot be recombined due to the fact that **s-cons** is not left-linear. Since **s-cons** essentially implements η-equivalence of substitutions we propose to turn the rule around as $s \rightarrow 1[s]\,.\,(\uparrow \circ s)$. Without any type-directed restrictions this rule is of course trivially non-terminating, but we conjecture that all that is needed to keep termination is the length of the context being substituted for. This information can easily be maintained through simple Church-style annotations: in the style of [Pfe08] we keep length-of-contexts as an intrinsic property of terms and substitutions and all other potential typing extrinsic.

With intrinsic information about context-lengths we can also add the η-rule $s \rightarrow \uparrow^n$ that applies whenever s occurs in a context of length n and substitutes for a context of length 0, as $\uparrow^n$ should be the only such substitution.

Let S denote a substitution meta-variable. Now, with a reverse **s-cons** $S \circ s$ is no longer a normal form but can expand to $1[S \circ s]\,.\,2[S \circ s]\,.\,3[S \circ s] \ldots \uparrow^m$. The individual entries $n[S \circ s]$ correspond to term meta-variables $X_n[s]$ and thus we have reduced the need for substitution meta-variables to term meta-variables for which confluence **(MC)** is already established.

5 Even Stronger Normalization

So far we have achieved strong normalization by disallowing β-steps in closures. With Theorem 12 we showed that substitutions are strongly normalizing, and it is therefore plausible that we could allow a single application of the unrestricted congruence rule in each step without breaking strong normalization. The intuition is that whenever we take a step inside a substitution we maintain the overall structure of the term disregarding the contents of substitutions. Thus, a nested induction on the strong normalization of the term and the strong normalization of all substitutions occurring within the term could presumably extend the strong normalization to this slightly more general reduction relation.

Going further, we can consider an extension of the reduction relation that allows the unrestricted congruence rule at most k times in each step for some fixed number k. As we saw in the counter-example, the non-terminating reduction sequence involved deeper and deeper nesting of **beta**-steps using the unrestricted closure congruence rule. This means that having such a fixed limit k is still going to rule out this particular counter-example.

In this section we formalize these ideas and prove the extended reduction relation strongly normalizing on well-typed terms and substitutions, thereby pushing the boundary of strong normalization right up to the limit set by the counter-example. We will in the following assume that every term and substitution is well-typed.

beta	$(\lambda M)\ N$	$\xrightarrow{k} M[N \,.\, \mathsf{id}]$
clos-const	$c[s]$	$\xrightarrow{k} c$
clos-var-dot1	$1[M \,.\, s]$	$\xrightarrow{k} M$
clos-var-dot2	$(n+1)[M \,.\, s]$	$\xrightarrow{k} n[s]$
clos-var-shift	$n[\uparrow^m]$	$\xrightarrow{k} n+m$
clos-clos	$M[s][t]$	$\xrightarrow{k} M[s \circ t]$
clos-lam	$(\lambda M)[s]$	$\xrightarrow{k} \lambda(M[1 \,.\, (s \circ \uparrow^1)])$
clos-app	$(M\ N)[s]$	$\xrightarrow{k} M[s]\ N[s]$
comp-id-L	$\uparrow^0 \circ s$	$\xrightarrow{k} s$
comp-cons	$(M \,.\, s) \circ t$	$\xrightarrow{k} M[t] \,.\, (s \circ t)$
comp-shift-dot	$\uparrow^{n+1} \circ (M \,.\, s)$	$\xrightarrow{k} \uparrow^n \circ s$
comp-shift-shift	$\uparrow^n \circ \uparrow^m$	$\xrightarrow{k} \uparrow^{n+m}$
comp-comp	$(s_1 \circ s_2) \circ s_3$	$\xrightarrow{k} s_1 \circ (s_2 \circ s_3)$
eta-subst	$(n+1) \,.\, \uparrow^{n+1}$	$\xrightarrow{k} \uparrow^n$

$$\frac{M \xrightarrow{k} M'}{M \,.\, s \xrightarrow{k} M' \,.\, s} \qquad \frac{s \xrightarrow{k} s'}{M \,.\, s \xrightarrow{k} M \,.\, s'} \qquad \frac{s \xrightarrow{k} s'}{s \circ t \xrightarrow{k} s' \circ t} \qquad \frac{t \xrightarrow{k} t'}{s \circ t \xrightarrow{k} s \circ t'}$$

$$\frac{M \xrightarrow{k} M'}{\lambda M \xrightarrow{k} \lambda M'} \qquad \frac{M \xrightarrow{k} M'}{M\ N \xrightarrow{k} M'\ N} \qquad \frac{N \xrightarrow{k} N'}{M\ N \xrightarrow{k} M\ N'}$$

$$\frac{M \xrightarrow{k} M'}{M[s] \xrightarrow{k} M'[s]} \qquad \frac{s \xrightarrow{k} s'}{M[s] \xrightarrow{k+1} M[s']}\ \textbf{clos1} \qquad \frac{s \to_\sigma s'}{M[s] \xrightarrow{0} M[s']}\ \textbf{clos2}$$

Fig. 5. Reduction rules

We will define a reduction relation $\xrightarrow{k}$ for each natural number $k \geq 0$ that allows k applications of the unrestricted congruence rule.

The reduction relation $\xrightarrow{k}$ is shown in its entirety in Figure 5. Notice that, for $k = 0$ the relation coincides with our regular reduction relation $\xrightarrow{0} = \to$, and of course a larger value of k allows more reductions $\xrightarrow{k} \subset \xrightarrow{k+1}$.

We will denote the subrelation of $\xrightarrow{k}$ that uses one of the rules **clos1** or **clos2** at least once as $\xrightarrow{k}_{[]}$ (read $\xrightarrow{k}_{[]}$ as "k-step-closure", the subscript closure brackets $[]$ should remind you that the reduction must happen inside a closure.) The subrelation that uses neither **clos1** nor **clos2** is denoted $\xrightarrow{k}_{\not[]}$ (read $\xrightarrow{k}_{\not[]}$ as "k-step-no-closure", the subscript crossed-over closure brackets $\not[]$ should remind you that the reduction must *not* happen inside a closure), such that $\xrightarrow{k} = \xrightarrow{k}_{[]} \cup \xrightarrow{k}_{\not[]}$. Since $\xrightarrow{k}_{\not[]}$ is independent of the value of k we will also write this relation as $\to_{\not[]}$.

We prove $\xrightarrow{k}$ strongly normalizing for terms and substitutions by induction on k. For $k = 0$ the relation is equal to $\to$ and therefore strongly normalizing by Theorems 11 and 12. In the following we will prove the induction step.

If we first consider $\xrightarrow{k+1}_{[]}$ then every step uses **clos1**. And this relation is therefore strongly normalizing by the simultaneous induction on the strong normalization of every substitution occurring inside the term, since these substitutions are in turn strongly normalizing by the induction on k.

Theorem 13 (`strong-ext.thm:is_sn1_clo,is_sn1s_clo`). *Assuming that $\xrightarrow{k}$ is strongly normalizing on substitutions, the relation $\xrightarrow{k+1}_{[]}$ is strongly normalizing on terms and substitutions.*

Assume we have $M_1 \xrightarrow{k+1}_{[]}^* M_2$ and we wish to prove M_2 strongly normalizing with respect to $\xrightarrow{k+1}$. Any $\xrightarrow{k+1}$ step taken by M_2 is either a $\xrightarrow{k+1}_{[]}$ step or a $\xrightarrow{k+1}_{\not[]}$ step. In the first case we again have $M_1 \xrightarrow{k+1}_{[]}^* M_2'$ with the same M_1, and this case cannot happen indefinitely since $\xrightarrow{k+1}_{[]}$ is strongly normalizing. In the second case we have $M_1 \xrightarrow{k+1}_{[]}^* M_2 \to_{\not[]} M_2'$. Since the steps from M_1 to M_2 only occur inside substitutions and the step from M_2 to M_2' only occurs outside substitutions, the idea is to prove that these two parts commute in some sense. That is, if we could prove $M_1 \to M_1' \xrightarrow{k+1}_{[]}^* M_2'$ then we could appeal to induction on the strong normalization of M_1 with respect to $\to$. Unfortunately this is not always the case, but this idea will lead us to something similar, which in the end will get us there.

One of the hard cases turns out to be $1[s_1] \xrightarrow{k+1}{}^*_{[]} 1[M_2' \,.\, s_2] \to_{/\!/} M_2'$ since this only gives us $s_1 \xrightarrow{k}{}^* M_2' \,.\, s_2$ to work with. To deal with this case (and a few similar cases) we will introduce a weak head normalization function for substitution compositions.

The functions $\mathsf{wcomp}(s)$ and $\mathsf{wcomp}(n; s)$ compute a weak head normal form of s and $\uparrow^n \circ s$, respectively. They are defined as follows and easily seen to be total.

$$\begin{array}{ll}
\mathsf{wcomp}(s) & = \mathsf{wcomp}(0; s) \\
\mathsf{wcomp}(n_1; \uparrow^{n_2}) & = \uparrow^{n_1+n_2} \\
\mathsf{wcomp}(0; M \,.\, s) & = M \,.\, s \\
\mathsf{wcomp}(n+1; M \,.\, s) & = \mathsf{wcomp}(n; s) \\
\mathsf{wcomp}(n; s_1 \circ s_2) & = \textbf{case}\ \mathsf{wcomp}(n; s_1)\ \textbf{of} \uparrow^{n'} \Rightarrow \mathsf{wcomp}(n'; s_2) \\
& \qquad\qquad\qquad\qquad\quad M \,.\, s \Rightarrow M[s_2] \,.\, (s \circ s_2)
\end{array}$$

Since wcomp plays a bit with the associativity of composition, the following theorems are not entirely trivial, but nevertheless true.

Theorem 14 (`strong-ext.thm:wcomp_to_msteps_su0`). *If* $\mathsf{wcomp}(n; s) = s'$ *then* $\uparrow^n \circ s \to^*_\sigma s'$.

Theorem 15 (`strong-ext.thm:wcomp0_to_msteps_su0`). *If* $\mathsf{wcomp}(s) = s'$ *then* $s \to^*_\sigma s'$.

The definition of wcomp is designed to do as little as possible. In particular, for $s_1 \circ s_2$ it avoids any reduction in s_2 whenever possible. This means that wcomp computes the least possibly reduced weak head normal form in a sense made precise by the following theorem. In particular, any reduction to a substitution s_2 with $\mathsf{wcomp}(s_2) = s_2$ factors through wcomp.

Theorem 16 (`strong-ext.thm:commute_wcomp_mstep1`). *If* $s_1 \xrightarrow{m}{}^* s_2$ *then* $\mathsf{wcomp}(n; s_1) \xrightarrow{m+1}{}^* \mathsf{wcomp}(n; s_2)$.

Returning our attention to the case $1[s_1] \xrightarrow{m}{}^*_{[]} 1[M \,.\, s_2] \to_{/\!/} M$, we can apply Theorem 16 to $s_1 \xrightarrow{m}{}^* M \,.\, s_2$ and thereby get a $\to$ reduction step of $1[s_1]$ to some M' with $M' \xrightarrow{m+1}{}^* M$. This deals with most of the otherwise problematic cases and allows us to prove the following theorem.

Theorem 17 (`strong-ext.thm:commute_clo_noc`). *If* $M_1 \xrightarrow{m}{}^*_{[]} M_2 \to_{/\!/} M_3$ *then there exists an* M *such that* $M_1 \to^+ M \xrightarrow{m+1}{}^* M_3$.

Theorem 17 presents the following diagram:

$$\begin{array}{ccc}
M_1 & \xrightarrow{\quad m \quad}{}^*_{[]} & M_2 \\
\big\downarrow & & \big\downarrow{\scriptstyle /\!/} \\
M & \xrightarrow{\quad m+1 \quad}{}^+ & M_3
\end{array}$$

Unfortunately, the bottom arrow is a general reduction sequence $M \xrightarrow{m+1}{}^* M_3$ and not a reduction sequence inside closures $M \xrightarrow{m+1}{}^*_{[]} M_3$. The reduction sequence from M to M_3 can be divided into $\xrightarrow{m+1}_{[]}$ steps and $\to_{/\!/}$ steps, so if it is not entirely consisting of $\xrightarrow{m+1}_{[]}$ steps, we can split it as $M \xrightarrow{m+1}{}^*_{[]} M_4 \to_{/\!/} M_5 \xrightarrow{m+1}{}^* M_3$ and apply Theorem 17 to the left half:

$$\begin{array}{ccccc}
M & \xrightarrow{\quad m+1 \quad}{}^*_{[]} & M_4 & & \\
\big\downarrow & & \big\downarrow{\scriptstyle /\!/} & & \\
M' & \xrightarrow{\quad m+2 \quad}{}^+ & M_5 & \xrightarrow{\quad m+1 \quad}{}^+ & M_3
\end{array}$$

We can repeat this construction on $M' \xrightarrow{m+2}{}^* M_3$ and since M is strongly normalizing with respect to $\to$ we will eventually reach $M \to^+ M'' \xrightarrow{m'}{}^*_{[]} M_3$ for some m'. Thus, we have strengthened Theorem 17 into:

Theorem 18 (`strong-ext.thm:commute_clo_noc2`). *If* $M_1 \xrightarrow{m}{}^*_{[]} M_2 \to_{[\!]} M_3$ *then there exists* m' *and* M *such that* $M_1 \to^+ M \xrightarrow{m'}{}^*_{[]} M_3$.

Now we can prove $\xrightarrow{k+1}$ strongly normalizing for some given term M_2 by a nested induction on the strong normalization of M_1 with respect to $\to$ and the strong normalization of M_2 with respect to $\xrightarrow{k+1}_{[]}$ and the invariant $M_1 \xrightarrow{m}{}^*_{[]} M_2$ for some m.

Theorem 19 (`strong-ext.thm:is_sn1`). *If* $\xrightarrow{k+1}_{[]}$ *is strongly normalizing then* $\xrightarrow{k+1}$ *is strongly normalizing for terms.*

Is it easy to adjust the proofs of Lemma 7 and Theorem 12 to yield strong normalization for substitutions with respect to $\xrightarrow{k+1}$ given strong normalization for terms.

With Theorem 13 and Theorem 19 we now have all the pieces to finish the induction on k and prove $\xrightarrow{k}$ strongly normalizing for all k.

Theorem 20 (`strong-ext.thm:strong_N`). *The reduction relation* $\xrightarrow{k}$ *is strongly normalizing for well-typed terms and substitutions for all* $k \geq 0$.

6 Conclusion

We have shown how a small restriction of the congruence rules of $\lambda\sigma$ yields a strongly normalizing calculus. We have shown that under this restriction, the $\lambda\sigma$-calculus satisfies the six desired properties **C**, **MC**, **PSN**, **SN**, **SIM**, and **FC**, and also the property **L**, i.e. no reduction rule relies on side conditions that inspect arbitrarily large terms. In addition to general insight into the normalization properties of explicit substitution calculi, this result also provides a very flexible foundation for the design of normalization procedures in any λ-calculus-based implementation, e.g. logical frameworks and proof assistants.

References

[Abe] Girard's proof of strong normalization of the simply-typed lambda-calculus. `http://abella.cs.umn.edu/examples/`.

[BR95] Roel Bloo and Kristoffer H. Rose. Preservation of Strong Normalisation in Named Lambda Calculi with Explicit Substitution and Garbage Collection. In *Computer Science in the Netherlands (CSN'95)*, pages 62–72, 1995.

[CHL96] P.-L. Curien, T. Hardin, and J.-J. Lévy. Confluence Properties of Weak and Strong Calculi of Explicit Substitutions. *Journal of the ACM (JACM)*, 43(2):362–397, 1996.

[CHR92] P.-L. Curien, T. Hardin, and A. Ríos. Strong Normalization of Substitutions. *Mathematical Foundations of Computer Science 1992*, pages 209–217, 1992.

[Kes07] Delia Kesner. The theory of calculi with explicit substitutions revisited. In *Computer Science Logic*, pages 238–252. Springer, 2007.

[Kes09] Delia Kesner. A Theory of Explicit Substitutions with Safe and Full Composition. *LMCS*, 2009.

[Mel95] Paul-André Mellies. Typed λ-calculi with explicit substitutions may not terminate. *Typed Lambda Calculi and Applications*, pages 328–334, 1995.

[NPP08] Aleksandar Nanevski, Frank Pfenning, and Brigitte Pientka. Contextual modal type theory. *ACM Trans. Comput. Logic*, 9:23:1–23:49, June 2008.

[Pfe08] Frank Pfenning. Church and Curry: Combining intrinsic and extrinsic typing. In *Reasoning in Simple Type Theory: Festschrift in Honor of Peter B. Andrews on His 70th Birthday*, Studies in Logic 17, pages 303–338. College Publications, 2008.

[SNS10] Anders Schack-Nielsen and Carsten Schürmann. Curry-Style Explicit Substitutions for the Linear and Affine Lambda Calculus. In *International Joint Conference on Automated Reasoning (IJCAR)*, pages 1–14, Edinburgh, UK, 2010. Springer LNCS 6173.