Metaconfluence of Calculi with Explicit Substitutions at a Distance*

Flávio L. C. de Moura[†1], Delia Kesner[2], and Mauricio Ayala-Rincón[1,3]

1 **Departamento de Ciência da Computação, Universidade de Brasília**
2 **Université Paris-Diderot, SPC, PPS, CNRS**
3 **Departamento de Matemática, Universidade de Brasília**

Abstract

Confluence is a key property of rewriting calculi that guarantees uniqueness of normal-forms when they exist. *Metaconfluence* is even more general, and guarantees confluence on *open/meta* terms, *i.e.* terms with holes, called *metavariables* that can be filled up with other (open/meta) terms. The difficulty to deal with open terms comes from the fact that the structure of metaterms is only *partially* known, so that some reduction rules became blocked by the metavariables. In this work, we establish metaconfluence for a family of calculi with explicit substitutions (ES) that enjoy preservation of strong-normalization (PSN) and that act *at a distance*. For that, we first extend the notion of reduction on metaterms in such a way that explicit substitutions are never structurally moved, *i.e.* they also act at a distance on metaterms. The resulting reduction relations are still rewriting systems, *i.e.* they do not include equational axioms, thus providing for the first time an interesting family of λ-calculi with explicit substitutions that enjoy both PSN and metaconfluence without requiring sophisticated notions of reduction modulo a set of equations.

1998 ACM Subject Classification F.4.1 Mathematical Logic: Lambda calculus and related systems

Keywords and phrases Confluence, Explicit Substitutions, Lambda Calculi

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2014.391

1 Introduction

Confluence is a key property of rewriting calculi that guarantees determinism of computations [9]. In confluent calculi, different reduction sequences starting at the same term always converge. When terms are enriched with *metavariables*, used to denote unknown parts of incomplete proofs/programs, we talk instead about *metaconfluence*, which in general does not follow directly from confluence.

In this paper we study metaconfluence of λ-calculi with explicit substitutions (ES), which are extensions of the λ-calculus being able to internalize the substitution operation [1, 20, 17, 13, 4]. Such calculi are used to refine/implement the notion of β-reduction in functional languages like Ocaml and Haskell, and proof-assistants like Coq, Isabelle, λProlog and PVS. Metaterms are notably introduced in this framework to implement higher-order unification and matching [16, 15, 8, 14] in proof-assistants. Indeed, a logical computational language based on higher-order resolution might be implemented using a calculus in which higher-order

* Work partially funded by the international project DeCOPA STIC-AmSud 146/2012.
† The first author was partially supported by FEMAT.

34th Int'l Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2014).
Editors: Venkatesh Raman and S. P. Suresh; pp. 391–402

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

unification can be treated in a natural way through the use of metavariables. Surprisingly, metaconfluence does not follows directly from confluence. Indeed, when the ES operator is propagated w.r.t. the structure of (meta)terms, then metavariables naturally block such propagations. The problem can be illustrated in terms of the following (simplified) rewriting system (we assume no capture of free variables holds):

$$\begin{array}{llll} (\lambda x.t)u & \mapsto & t[x/u] & (\lambda x.t)[y/u] \mapsto \lambda x.t[y/u] \\ (tv)[y/u] & \mapsto & t[y/u]v[y/u] & t[y/u] \mapsto t, \text{ if } y \text{ does not occur free in } t \end{array}$$

which generates the following diverging reduction sequences on metaterms:

$$t[y/u][x/v] \twoheadleftarrow ((\lambda x.t)v)[y/u] \to t[x/v][y/u]$$

where y does not occur free in v, $\to$ denotes the contextual closure of $\mapsto$ and $\twoheadrightarrow$ the reflexive-transitive closure of $\to$. Thus, there exist many λ-calculi with ES that are confluent but not metaconfluent. Some of them, as for example, $\lambda\sigma$ [1] and λs [19], were extended respectively to $\lambda\sigma_{\Uparrow}$ [12] and λs_e [20] in order to regain metaconfluence. Nevertheless, these extended calculi do not enjoy PSN [25, 18], thus showing the fragility of such rewriting systems.

Another solution was adopted by calculi with ES inspired from linear logic proof-nets [28], such as $\lambda\mathtt{es}$ [21] and $\lambda\mathtt{ex}$ [22]. Metaconfluence is recovered in these calculi by adding to the rewriting systems an equational axiom for commutation of independent substitutions:

$$t[y/u][x/v] \sim t[x/v][y/u]$$

where x (resp. y) does not occur free in u (resp. v). The resulting reduction calculi turned out to have very good properties, in particular, metaconfluence and PSN can live together [22]. However, equational reasoning becomes unavoidable, and even if commutation of independent substitutions is obtained for free when λ-terms are represented by proof-nets, this is not the case in classical implementations which use algebraic λ-terms.

In this paper we give a solution to this problem by pushing further the ES paradigm inspired from Linear-Logic Proof-Nets. In particular, there are nowadays several calculi, also inspired from Linear-Logic Proof-Nets, that are based on the idea that the ES operation acts *at a distance* and does not need to be percolated over the structure of terms. Typical examples of such calculi are the linear substitution (or Milner's) calculus and the structural lambda-calculus (see resp. [26] and [4]), that belong to this new paradigm and have been successfully used for different applications such as implicit complexity [7], the theory of abstract standardization [3], or abstract machines [2].

We prove metaconfluence for a family of calculi with ES that act *at a distance*, namely the substitution calculus [5], the linear substitution calculus [26] and the structural λ-calculus [4]. The resulting reduction systems are still simple rewriting systems, i.e. they do not include equational axioms. The key of our solution relies on a new notion of substitution that propagates/applies only those that are affecting real variables, by keeping fixed the ones affecting metavariables. For instance, if $\mathbb{Y}_x$ denotes a metavariable in a context having only the free variable x, then, according to our new notion of metasubstitution, the term $(\mathbb{Y}_x\ x)[x/u]$ reduces to $(\mathbb{Y}_x\ u)[x/u]$ and not to $\mathbb{Y}_x[x/u]u$ (which was the solution adopted in [24, 27]). The idea is that the *real* variable x of the metaterm $\mathbb{Y}_x\ x$ can be substituted by u, as usual, but the substitution $[x/u]$ remains fixed in its place and does not percolate the application, it must be delayed because of the metavariable $\mathbb{Y}_x$. This notion of metasubstitution turns out to be essential in the development of the results we show in this paper.

To establish metaconfluence for our three calculi, we use the Hindley-Rosen Theorem [10], which states that if two *confluent* rewriting systems *commute* then their *union is also*

confluent. To do so, each rewriting system we treat in this paper is split into two rewriting systems, say $\mathcal{R}$ and §. We then show that $\mathcal{R}$ and § alone are confluent. Finally, we show that $\mathcal{R}$ *strongly commutes* with §, *i.e.* $\forall t_0, t_1, t_2$, if $t_0 \to_{\mathcal{R}} t_1$ and $t_0 \to_{\S} t_2$, $\exists t_3$ s.t. $t_1 \twoheadrightarrow_{\S} t_3$ and $t_2 \stackrel{=}{\to}_{\mathcal{R}} t_3$ (*i.e.* $t_2 \to_{\mathcal{R}} t_3$ or $t_2 = t_3$). Since strongly commutation implies commutation [9], then we are done by Hindley-Rosen Theorem.

We thus provide an interesting set of λ-calculi with ES that act at a distance, which enjoy both PSN and metaconfluence without requiring sophisticated notions of reduction modulo a set of equations. Moreover, in contrast to available proofs of metaconfluence in the literature [12, 20, 27, 24], which are non-trivial, our approach just needs a simple reasoning about commutation of reductions, a standard notion used in abstract reduction systems [11].

2 Common Syntax for Terms

Explicit substitutions calculi with names are built over a simple grammar which is an extension of that of the λ-calculus:

$$t, u ::= x \mid t\ u \mid \lambda x.t \mid t[x/u] \tag{1}$$

The symbol x is called a *variable*, $\lambda x.t$ an *abstraction*, $t\ u$ an *application* and $t[x/u]$ a term with an *explicit substitution (ES)* $[x/u]$, *i.e.* a substitution waiting to be applied. The abstraction $\lambda x.t$ and the ES $t[x/u]$ both bind x in t. The notions of **free** and **bound** variables are defined as usual, in particular, $\mathtt{fv}(t[x/u]) := \mathtt{fv}(t) \setminus \{x\} \cup \mathtt{fv}(u)$, $\mathtt{fv}(\lambda x.t) := \mathtt{fv}(t) \setminus \{x\}$, $\mathtt{bv}(t[x/u]) := \mathtt{bv}(t) \cup \{x\} \cup \mathtt{bv}(u)$ and $\mathtt{bv}(\lambda x.t) := \mathtt{bv}(t) \cup \{x\}$. We work with the standard notion of α-conversion *i.e.* the bound variables can be renamed in order to avoid clashes with the free ones. Thus, terms are always considered modulo α-equivalence, *i.e.* we work on α-equivalence classes of terms. We use $|t|_x$ to denote the number of free occurrences of the variable x in the term t. When $|t|_x = n \geq 2$, we write $t_{\langle x|y\rangle}$ for the **non-deterministic replacement** of i $(1 \leq i \leq n-1)$ free occurrences of x in t by a *fresh* variable y. Thus for example, given $u = (x\ z)[z/x]$, we have $|u|_x = 2$ so that only one replacement of x in u can be done to construct $u_{\langle x|y\rangle}$, which then denotes either $(y\ z)[z/x]$ or $(x\ z)[z/y]$ but not $(y\ z)[z/y]$. Contexts are defined as usual, *i.e.* they are given by the following grammar:

$$C ::= \Box \mid Ct \mid tC \mid \lambda x.C \mid C[x/t] \mid t[x/C]$$

We write $C[t]$ for the term obtained by replacing the hole $\Box$ of C by t, thus *e.g.* $(\Box y)[x] = xy$ and $(\lambda x.\Box)[x] = \lambda x.x$. We write $C[\![u]\!]$ when the free variables of u are not captured by the context C, thus for example, $C[\![x]\!]$ denotes the term xy if $C = \Box y$, and $\lambda y.x$, if $C = \lambda x.\Box$.

Substitutions are (finite) functions from variables to terms. We denote a non-empty substitution σ by $\{x_1/u_1, \ldots, x_n/u_n\}$ $(n \geq 1)$ and the empty substitution by $\mathtt{Id}$. The domain of the substitution σ is given by $\mathtt{dom}(\sigma) := \{x \mid \sigma(x) \neq x\}$. The set $\mathtt{var}(\sigma)$ is given by $\cup_{x \in \mathtt{dom}(\sigma)} \mathtt{fv}(\sigma(x))$. The **application** of a **substitution** σ to a term t is defined by induction on the structure of terms as follows:

$$\begin{array}{llll} x\sigma := \sigma(x) & \text{if } x \in \mathtt{dom}(\sigma) & (\lambda y.t)\sigma := \lambda y.t\sigma & \text{if } y \notin \mathtt{var}(\sigma) \\ y\sigma := y & \text{if } y \notin \mathtt{dom}(\sigma) & t[y/u]\sigma := t\sigma[y/u\sigma] & \text{if } y \notin \mathtt{var}(\sigma) \\ (tu)\sigma := (t\sigma)(u\sigma) & & & \end{array}$$

Here it should be stressed that the third and fourth rules are conceived modulo α-conversion. Thus for example $(\lambda y.x)\{x/y\} = \lambda z.y$. Remark that $t\{x/u\} = t$ if $x \notin \mathtt{fv}(t)$.

We now present the reduction rules of three calculi with ES acting at a distance that are based on grammar (1). The first one, known as the substitution calculus, splits the non-terminating β-rule of the λ-calculus into two terminating rules $\mathtt{dB}$ and $\mathtt{s}$ (see Fig. 1).

$$\begin{array}{lll} (\lambda x.t)\mathtt{L}\ u & \mapsto_{\mathtt{dB}} & t[x/u]\mathtt{L} \\ t[x/u] & \mapsto_{\mathtt{s}} & t\{x/u\} \end{array}$$

Figure 1 The substitution calculus for terms.

$$\begin{array}{lll} (\lambda x.t)\mathtt{L}\ u & \mapsto_{\mathtt{dB}} & t[x/u]\mathtt{L} \\ \mathtt{C}[\![x]\!][x/u] & \mapsto_{\mathtt{ls}} & \mathtt{C}[\![u]\!][x/u] \\ t[x/u] & \mapsto_{\mathtt{w}} & t \qquad \text{if } |t|_x = 0 \end{array}$$

Figure 2 The linear substitution calculus for terms.

$$\begin{array}{lll} (\lambda x.t)\mathtt{L}\ u & \mapsto_{\mathtt{dB}} & t[x/u]\mathtt{L} \\ t[x/u] & \mapsto_{\mathtt{c}} & t_{\langle x|y\rangle}[x/u][y/u] \qquad \text{if } |t|_x > 1 \\ t[x/u] & \mapsto_{\mathtt{d}} & t\{x/u\} \qquad \text{if } |t|_x = 1 \\ t[x/u] & \mapsto_{\mathtt{w}} & t \qquad \text{if } |t|_x = 0 \end{array}$$

Figure 3 The structural substitution calculus for terms.

The L symbol appearing in the dB-rule, also called d*istant* B*eta*, denotes a (possibly empty) list of substitutions of the form $[x_1/t_1][x_2/t_2]\dots[x_n/t_n]$ $(n \geq 0)$ [1] The resulting reduction relation, obtained by the contextual closure of the rewriting rules, is written $\to_{\lambda_{\mathtt{sub}}}$.

The second calculus (see Fig. 2) is known as the linear substitution calculus. Rule $\mapsto_{\mathtt{dB}}$ (resp. $\mapsto_{\mathtt{ls}}$) comes from the structural λ-calculus [4] (resp. Milner's calculus [26]), while $\mapsto_{\mathtt{w}}$ belongs to both calculi. The calculus performs *partial* substitution in the sense that only one free variable occurrence is substituted at a time. This partial substitution, performed by means of the ls-rule (for linear substitution), is non-deterministic, *i.e.* ls randomly chooses the free variable occurrence of x to be substituted by u. The resulting reduction relation, obtained by the contextual closure of the rewriting rules, is written $\to_{\lambda_{\mathtt{lsub}}}$.

The third calculus (see Fig. 3) is the structural λ-calculus [4]. The dB-rule triggers computation. The c-rule duplicates ES which affect a term $t_{\langle x|y\rangle}$, denoting, as defined above, some non-deterministic replacement of a non-empty subset of the free occurrences of the variable x in t by a fresh variable y. Metasubstitution on variables is only performed by the d-rule where the variables have only one single occurrence in the term. The resulting reduction relation, obtained by the contextual closure of the rewriting rules, is written $\to_{\lambda_{\mathtt{str}}}$.

In what follows we denote by $\to_r$ (resp. $\twoheadrightarrow_r$) the contextual (resp. reflexive-transitive) closure of each rewriting rule $\mapsto_r$ (resp. reduction relation $\to_r$), for $r \in \{\mathtt{dB}, \mathtt{s}, \mathtt{ls}, \mathtt{w}, \mathtt{c}, \mathtt{d}\}$ introduced before. For each calculus, the reduction relation associated to its substitution calculus, *i.e.* generated by all its rewriting rules except $\mapsto_{\mathtt{dB}}$, are defined by $\to_{\mathtt{s}} := \to_{\lambda_{\mathtt{sub}}} \setminus \to_{\mathtt{dB}}$, $\to_{\mathtt{lsub}} := \to_{\lambda_{\mathtt{lsub}}} \setminus \to_{\mathtt{dB}}$ and $\to_{\mathtt{str}} := \to_{\lambda_{\mathtt{str}}} \setminus \to_{\mathtt{dB}}$ respectively. A reduction relation $\to_{\mathcal{R}}$ is said to be **confluent** on terms (resp. metaterms) iff for all terms (resp. metaterms) t_0, t_1, t_2, if $t_0 \twoheadrightarrow_{\mathcal{R}} t_1$ and $t_0 \twoheadrightarrow_{\mathcal{R}} t_2$, there exists a term (resp. metaterm) t_3 s.t. $t_1 \twoheadrightarrow_{\mathcal{R}} t_3$ and $t_2 \twoheadrightarrow_{\mathcal{R}} t_3$.

Here are some examples of reduction sequences from the term $t = (\lambda x.xx)y$:

In $\lambda_{\mathtt{sub}}$: $t \to_{\mathtt{dB}} (xx)[x/y] \to_{\mathtt{s}} yy$

In $\lambda_{\mathtt{lsub}}$: $t \to_{\mathtt{dB}} (xx)[x/y] \to_{\mathtt{ls}} (xy)[x/y] \to_{\mathtt{ls}} (yy)[x/y] \to_{\mathtt{w}} yy$

In $\lambda_{\mathtt{str}}$: $t \to_{\mathtt{dB}} (xx)[x/y] \to_{\mathtt{c}} (xx')[x/y][x'/y] \to_{\mathtt{d}} (yx')[x'/y] \to_{\mathtt{d}} yy$

[1] Formally, the list L is a context generated by the grammar $\Box \mid \mathtt{L}[x/t]$.

All the calculi presented above, let us write $\mathcal{R}$, enjoy good properties, specially: (*simulation*) every β-reduction step in the λ-calculus can be performed in $\mathcal{R}$, (*confluence*) all divergent reduction sequences in $\mathcal{R}$ can be closed and (*preservation of β-strong normalization*) every β-strongly normalizing λ-term is also $\mathcal{R}$-strongly normalizing.

3 Common Syntax for MetaTerms

We now extend the grammar of terms to metaterms by adding metavariables which denote incomplete proofs/programs in higher-order theories. In particular, metavariables are used in higher-order unification to denote unkown partial solutions to be instantiated by the unification procedure [15, 8, 14]. We use $\mathbb{X}_\Delta$ to denote a metavariable with free variables in the set Δ. The grammar (1) introduced in Sec. 2 is then extended as follows:

$$t, u ::= x \mid \mathbb{X}_\Delta \mid t\ u \mid \lambda x.t \mid t[x/u] \tag{2}$$

We also extend the notation $|t|_x$ to metaterms, thus *e.g.* $|(y\mathbb{X}_y)[z/\mathbb{Z}_y]|_y = 3$. We distinguish between **free meta** and **real** variables. They are both defined by induction as follows.

$$\begin{array}{lcllcl}
\mathtt{fm}(x) & := & \emptyset & \mathtt{fr}(x) & := & \{x\} \\
\mathtt{fm}(\mathbb{X}_\Delta) & := & \Delta & \mathtt{fr}(\mathbb{X}_\Delta) & := & \emptyset \\
\mathtt{fm}(tu) & := & \mathtt{fm}(t) \cup \mathtt{fm}(u) & \mathtt{fr}(tu) & := & \mathtt{fr}(t) \cup \mathtt{fr}(u) \\
\mathtt{fm}(\lambda x.t) & := & \mathtt{fm}(t) \setminus \{x\} & \mathtt{fr}(\lambda x.t) & := & \mathtt{fr}(t) \setminus \{x\} \\
\mathtt{fm}(t[x/u]) & := & \mathtt{fm}(t) \setminus \{x\} \cup \mathtt{fm}(u) & \mathtt{fr}(t[x/u]) & := & \mathtt{fr}(t) \setminus \{x\} \cup \mathtt{fr}(u)
\end{array}$$

Thus for example, given $t = (\mathbb{X}_{\{x,y\}} z)[x/\mathbb{Y}_\emptyset]$ we have $\mathtt{fm}(t) = \{y\}$ and $\mathtt{fr}(t) = \{z\}$. The set of **free** variables of a metaterm is given by $\mathtt{fv}(t) := \mathtt{fm}(t) \cup \mathtt{fr}(t)$. We extend the non-deterministic operation $\_\langle\_|\_\rangle$ introduced in Sec. 2 to metaterms as expected.

For each λ-calculus in this paper, the metaconfluence proof uses the termination property of the corresponding substitution subcalculus. The first substitution calculus, given by the reduction relation $\to_{\mathtt{s}}$, is trivially terminating. In order to prove termination of the substitution subcalculus $\to_{\mathtt{lsub}} := \to_{\mathtt{ls}} \cup \to_{\mathtt{w}}$ we use a decreasing measure based on the notion of multiplicity [24]. Indeed, the **size** of a metaterm is recursively defined as follows:

$$\begin{array}{lcllcl}
\mathtt{sz}(x) = \mathtt{sz}(\mathbb{X}_\Delta) & := & 1 & \mathtt{sz}(t\ u) & := & \mathtt{sz}(t) + \mathtt{sz}(u) \\
\mathtt{sz}(\lambda x.t) & := & \mathtt{sz}(t) & \mathtt{sz}(t[x/u]) & := & \mathtt{sz}(t) + \mathtt{sz}(u) \cdot (1 + \mathtt{ml}_x(t))
\end{array}$$

where $\mathtt{ml}_x(t)$, the **multiplicity** of the variable x in the metaterm t, is defined by:

$$\mathtt{ml}_x(t) := 0 \qquad \text{if } x \notin \mathtt{fv}(t), \text{ otherwise}$$

$$\begin{array}{lcllcl}
\mathtt{ml}_x(x) = \mathtt{ml}_x(\mathbb{X}_\Delta) & := & 1 & \mathtt{ml}_x(t\ u) & := & \mathtt{ml}_x(t) + \mathtt{ml}_x(u) \\
\mathtt{ml}_x(\lambda y.t) & := & \mathtt{ml}_x(t) & \mathtt{ml}_x(t[y/u]) & := & \mathtt{ml}_x(t) + \mathtt{ml}_x(u) \cdot (1 + \mathtt{ml}_y(t))
\end{array}$$

We have for example $\mathtt{sz}((x\ x)[x/\lambda y.y]) = 5$, $\mathtt{sz}((x\ z)[x/\lambda y.y][z/\lambda y.y]) = 6$ and $\mathtt{sz}((z\ z)[z/x\ x][x/\lambda y.y]) = 15$. Observe that $\mathtt{sz}(t) \geq 1$ and $\mathtt{ml}_x(t) \geq 0$. Moreover, $x \notin \mathtt{fv}(t)$ implies $\mathtt{ml}_x(t) = 0$. It is easy to extend these measures to contexts by adding $\mathtt{sz}(\Box) = 0$ and $\mathtt{ml}_x(\Box) = 0$.

While this measure is decreasing for $\mathtt{lsub}$, *i.e.* $t \to_{\mathtt{lsub}} t'$ implies $\mathtt{sz}(t) > \mathtt{sz}(t')$ (see Sec. 4.1 for details), this is not the case for the subcalculus $\to_{\mathtt{str}} := \to_{\mathtt{c}} \cup \to_{\mathtt{d}} \cup \to_{\mathtt{w}}$. Thus for example, $(x\ x)[x/u] \to_{\mathtt{c}} (x\ z)[x/u][z/u]$ but $\mathtt{sz}((x\ x)[x/u]) < \mathtt{sz}((x\ z)[x/u][z/u])$. We then introduce another measure (cf. [4]) which will be used in Sec. 4.3 to show that $\to_{\mathtt{str}}$

terminates. In what follows $[\,]$ denotes the empty multiset, $\sqcup$ the multiset union and $n \cdot [a_1, \ldots, a_n]$ the multiset $[n \cdot a_1, \ldots, n \cdot a_n]$.

The **multimeasure** of a metaterm t, written $\mathtt{jm}(t)$, is a multiset of integers defined as:

$$\begin{array}{llllll} \mathtt{jm}(x) = \mathtt{jm}(\mathbb{X}_\Delta) & := & [\,] & \mathtt{jm}(t\ u) & := & \mathtt{jm}(t) \sqcup \mathtt{jm}(u) \\ \mathtt{jm}(\lambda x.t) & := & \mathtt{jm}(t) & \mathtt{jm}(t[x/u]) & := & [\mathtt{P}_x(t)] \sqcup \mathtt{jm}(t) \sqcup \max(1, \mathtt{P}_x(t)) \cdot \mathtt{jm}(u) \end{array}$$

where $\mathtt{P}_x(t)$ denotes the **potential multiplicity** of the variable x in the term t and is recursively defined on α-equivalence classes of t as follows: $\mathtt{P}_x(t) := 0$ if $x \notin \mathtt{fv}(t)$, otherwise

$$\begin{array}{llllll} \mathtt{P}_x(x) = \mathtt{P}_x(\mathbb{X}_\Delta) & := & 1 & \mathtt{P}_x(t\ u) & := & \mathtt{P}_x(t) + \mathtt{P}_x(u) \\ \mathtt{P}_x(\lambda y.t) & := & \mathtt{P}_x(t) & \mathtt{P}_x(t[y/u]) & := & \mathtt{P}_x(t) + \max(1, \mathtt{P}_y(t)) \cdot \mathtt{P}_x(u) \end{array}$$

Note that in the second case, necessarily $x \in \Delta$. Thus for example, $\mathtt{jm}((x\ x)[x/\lambda y.y]) = [2]$; $\mathtt{jm}((x\ z)[x/\lambda y.y][z/\lambda y.y]) = [1,1]$ and $\mathtt{jm}((z\ z)[z/x\ x][x/\lambda y.y]) = [4,2]$.

4 Metaconfluence

This section is devoted to the proofs of metaconfluence of our three calculi. We start by extending the notion of metasubstitution introduced in Sec. 2 to metaterms. A first approach, already used in [23] for the $\lambda\mathtt{ex}$-calculus, is obtained by adding to the metasubstitution operation on terms the following case :

$$\mathbb{X}_\Delta\{x/u\} = \begin{cases} \mathbb{X}_\Delta[x/u] & \text{if } x \in \Delta \\ \mathbb{X}_\Delta & \text{otherwise.} \end{cases} \tag{3}$$

Nevertheless, if one naively uses this specification to extend the $\mathtt{s}$-rule to metaterms, termination is lost as the following example shows: $\mathbb{X}_{\{x\}}[x/u] \to_{\mathtt{s}} \mathbb{X}_{\{x\}}\{x/u\} = \mathbb{X}_{\{x\}}[x/u] \to_{\mathtt{s}} \ldots$ This can be recovered by simply restricting the form of the metaterms t on the left-hand side of the $\mathtt{s}$-rule to those that are not metavariables affected by ES (as done for example in [24]). However, even with this restriction, confluence fails:

$$X_{\{x,y\}}[x/u][y/v] \leftarrow (\lambda y.\mathbb{X}_{\{x,y\}})[x/u]\ v \to_{\mathtt{dB}} \mathbb{X}_{\{x,y\}}[y/v][x/u]$$

One can then add equations between metaterms to allow permutation of independent substitutions (cf. [21]) in order to close this divergent diagram. But then equational reasoning becomes necessary to deal with the resulting reduction system (a reduction system modulo); this could be particularly problematic from an implementation point of view.

In this paper we present another approach where no additional equations are necessary to guarantee metaconfluence. We start by extending the notion of metasubstitution to metaterms as follows. The (capture-free) **fixed metasubstitution** of x by the metaterm u in the metaterm t, written $t[\![x/u]\!]$, is given by:

$$t[\![x/u]\!] := \begin{cases} t\{\!\{x/u\}\!\}[x/u], & \text{if } x \in \mathtt{fm}(t) \\ t\{\!\{x/u\}\!\}, & \text{if } x \notin \mathtt{fm}(t) \end{cases}$$

where the operation $\_\{\!\{\_/\_\}\!\}$ is defined as follows: $t\{\!\{x/u\}\!\} := t$ if $x \notin \mathtt{fr}(t)$, otherwise

$$\begin{array}{llll} x\{\!\{x/u\}\!\} & := u; & (\lambda y.v)\{\!\{x/u\}\!\} := \lambda y.v\{\!\{x/u\}\!\} & (x \neq y\ \&\ y \notin \mathtt{fv}(u)); \\ (tv)\{\!\{x/u\}\!\} := t\{\!\{x/u\}\!\}v\{\!\{x/u\}\!\}; & & t[y/v]\{\!\{x/u\}\!\} := t\{\!\{x/u\}\!\}[y/v\{\!\{x/u\}\!\}] & (x \neq y\ \&\ y \notin \mathtt{fv}(u)). \end{array}$$

Thus for example, $(\lambda y.x\mathbb{X}_{\{x\}})[\![x/y]\!] = (\lambda z.y\mathbb{X}_{\{x\}})[x/y]$. Remark that renaming of the bound variable y was done to avoid capture of free variables. The *real* free occurrence of x

$(\lambda x.t)\mathtt{L}\ u$	$\mapsto_{\mathtt{dB}}$	$t[x/u]\mathtt{L}$	
$t[x/u]$	$\mapsto_{\mathtt{s}}$	$t[\![x/u]\!]$	if $x \notin \mathtt{fm}(t)$ or $x \in \mathtt{fr}(t)$

Figure 4 The substitution calculus for metaterms.

$(\lambda x.t)\mathtt{L}\ u$	$\mapsto_{\mathtt{dB}}$	$t[x/u]\mathtt{L}$	
$t[x/u]$	$\mapsto_{\mathtt{c}}$	$t_{\langle x\|y\rangle}[x/u][y/u]$	if $\|t\|_x > 1$
$t[x/u]$	$\mapsto_{\mathtt{d}}$	$t[\![x/u]\!]$	if $\|t\|_x = 1$ and $x \notin \mathtt{fm}(t)$
$t[x/u]$	$\mapsto_{\mathtt{w}}$	t	if $\|t\|_x = 0$

Figure 5 The structural lambda calculus for metaterms.

was substituted by y, however, the ES $[x/y]$ remains fixed in the resulting term since it is affecting a metavariable with scope x.

The above definition is a key notion of this work: it is able to capture metasubstitution on *terms*, but it is compatible with *metaterms*. Formally, the metasubstitution $\_[\![\_/\_]\!]$ is splited into two complementary notions of substitution: the ES $\_[\_/\_]$, which is fixed whenever there is a metavariable with scope in the domain of this substitution; and the implicit substitution $\_\{\!\{\_/\_\}\!\}$ on terms, that only acts on real variables.

We can now reformulate the first and the third λ-calculi presented before by using fixed metasubstitution $[\![x/u]\!]$ on *metaterms* instead of $\{x/u\}$ on *terms*. The resulting reduction relations are shown in Fig. 4 and Fig. 5, respectively. In the case of the linear substitution calculus, since the reduction relation on terms does not use metasubstitution, we can keep exactly the same rewriting rules in Fig. 2 to specify reduction on *metaterms*. In the three cases, the resulting reduction systems on metaterms are conservative w.r.t. their respective reduction notions on terms.

4.1 The Substitution Calculus enjoys MetaConfluence

This section presents the metaconfluence proof for the substitution calculus. We start by stating some useful properties concerning the notion of metasubstitution that will be important in the rest of this section.

▶ **Lemma 1.** *Let t, u, v be metaterms. If $x \neq y$ and $x \notin \mathtt{fv}(v)$ then $t\{\!\{x/u\}\!\}\{\!\{y/v\}\!\} = t\{\!\{y/v\}\!\}\{\!\{x/u\{\!\{y/v\}\!\}\}\!\}$ and $t[\![x/u]\!]\{\!\{y/v\}\!\} = t\{\!\{y/v\}\!\}[\![x/u\{\!\{y/v\}\!\}]\!]$.*

Proof. The first statement is by induction on t and the second one uses the first one. ◀

The next lemma states that the substitution calculus on metaterms is stable w.r.t. the new notion of metasubstitution.

▶ **Lemma 2** (Stability). *Let t, u be metaterms. Let $r \in \{\mathtt{dB}, \mathtt{s}\}$.*

- *If $t \to_r t'$, then $t\{\!\{x/u\}\!\} \to_r t'\{\!\{x/u\}\!\}$ and $t[\![x/u]\!] \to_r t'[\![x/u]\!]$.*
- *If $u \to_r u'$, then $t\{\!\{x/u\}\!\} \twoheadrightarrow_r t\{\!\{x/u'\}\!\}$ and $t[\![x/u]\!] \twoheadrightarrow_r t[\![x/u']\!]$.*

Proof.

- The first statement is by induction on $t \to_{\lambda_{\mathtt{sub}}} t'$ using Lem. 1. The second one uses the first one.
- The first statement is by induction on $u \to_{\lambda_{\mathtt{sub}}} u'$ and the second one by using the first one. ◀

The stability properties are necessary to prove that the reduction systems $\to_{\mathtt{dB}}$ and $\to_{\mathtt{s}}$ strongly commutes.

▶ **Theorem 3** (Strong Commutation). *$\forall t_0, t_1, t_2$, if $t_0 \to_{\mathtt{s}} t_1$ and $t_0 \to_{\mathtt{dB}} t_2$, $\exists t_3$ s.t. $t_1 \twoheadrightarrow_{\mathtt{dB}} t_3$ and $t_2 \overset{=}{\to}_{\mathtt{s}} t_3$.*

Proof. By induction on the reduction relation. We only show here the key cases:

$$\begin{array}{ccc} t[x/u] \to_{\mathtt{s}} t[\![x/u]\!] & t[x/u] \to_{\mathtt{s}} t[\![x/u]\!] & (\lambda x.t)\mathtt{L}u \to_{\mathtt{s}} (\lambda x.t')\mathtt{L}u \\ \downarrow_{\mathtt{dB}} \quad \downarrow_{\mathtt{dB}}\ (Lem.\ 2) & \downarrow_{\mathtt{dB}} \quad \downarrow_{\mathtt{dB}} (Lem.\ 2) & \downarrow_{\mathtt{dB}} \quad \downarrow_{\mathtt{dB}} \\ t'[x/u] \to_{\mathtt{s}} t'[\![x/u]\!] & t[x/u'] \to_{\mathtt{s}} t[\![x/u']\!] & t[x/u]\mathtt{L} \to_{\mathtt{s}} t'[x/u]\mathtt{L} \end{array}$$

$$\begin{array}{ccc} (\lambda x.t)\mathtt{L}u \to_{\mathtt{s}} (\lambda x.t)\mathtt{L}'u & (\lambda x.t)\mathtt{L}u \to_{\mathtt{s}} (\lambda x.t)\mathtt{L}u' & (\lambda x.t)\mathtt{L}_1[y/v]\mathtt{L}_2 u \to_{\mathtt{s}} (\lambda x.t)\mathtt{L}_1[\![y/v]\!]\mathtt{L}_2 u \\ \downarrow_{\mathtt{dB}} \quad \downarrow_{\mathtt{dB}} & \downarrow_{\mathtt{dB}} \quad \downarrow_{\mathtt{dB}} & \downarrow_{\mathtt{dB}} \quad \downarrow_{\mathtt{dB}} \\ t[x/u]\mathtt{L} \to_{\mathtt{s}} t[x/u]\mathtt{L}' & t[x/u]\mathtt{L} \to_{\mathtt{s}} t[x/u']\mathtt{L} & t[x/u]\mathtt{L}_1[y/v]\mathtt{L}_2 \to_{\mathtt{s}} t[x/u]\mathtt{L}_1[\![y/v]\!]\mathtt{L}_2 \end{array}$$

◀

We can now conclude metaconfluence of the substitution calculus as follows:

▶ **Corollary 4.** *The reduction relation $\to_{\lambda_{\mathtt{sub}}}$ is confluent on metaterms.*

Proof. Let $\to_{\lambda_{\mathtt{sub}}} := \to_{\mathtt{dB}} \cup \to_{\mathtt{s}}$. Both $\to_{\mathtt{dB}}$ and $\to_{\mathtt{str}}$ are trivially confluent. They commute (Theorem 3). We conclude by the Hindley-Rosen Theorem [10] introduced in Sec 1. ◀

4.2 The Linear Substitution Calculus enjoys MetaConfluence

In this section we prove metaconfluence for the linear substitution calculus. As in the previous section, we first prove that the systems $\to_{\mathtt{dB}}$ and $\to_{\mathtt{lsub}}$ strongly commute, where, as defined in Sec. 3 we have $\to_{\mathtt{lsub}} := \to_{\mathtt{ls}} \cup \to_{\mathtt{w}}$.

▶ **Theorem 5** (Strong Commutation). *$\forall t_0, t_1, t_2$, if $t_0 \to_{\mathtt{lsub}} t_1$ and $t_0 \to_{\mathtt{dB}} t_2$, $\exists t_3$ s.t. $t_1 \twoheadrightarrow_{\mathtt{dB}} t_3$ and $t_2 \overset{=}{\to}_{\mathtt{lsub}} t_3$.*

Proof. By induction on the reduction relations, then, for the base cases, by case analysis of overlapping local divergences. Most of the cases are straightforward, we only show here the more interesting ones.

$$\begin{array}{cc} \mathtt{C}[\![x]\!][x/u] \to_{\mathtt{ls}} \mathtt{C}[\![u]\!][x/u] & (\lambda x.\mathtt{C}[\![y]\!])\mathtt{L}_1[y/v]\mathtt{L}_2 u \to_{\mathtt{ls}} (\lambda x.\mathtt{C}[\![v]\!])\mathtt{L}_1[y/v]\mathtt{L}_2 u \\ \downarrow_{\mathtt{dB}} \quad \downarrow_{\mathtt{dB}} & \downarrow_{\mathtt{dB}} \quad \downarrow_{\mathtt{dB}} \\ \mathtt{C}'[\![x]\!][x/u] \to_{\mathtt{ls}} \mathtt{C}'[\![u]\!][x/u] & \mathtt{C}[\![y]\!][x/u]\mathtt{L}_1[y/v]\mathtt{L}_2 \to_{\mathtt{ls}} \mathtt{C}[\![v]\!][x/u]\mathtt{L}_1[y/v]\mathtt{L}_2 \end{array}$$

$$\begin{array}{cc} \mathtt{C}[\![x]\!][x/u] \to_{\mathtt{ls}} \mathtt{C}[\![u]\!][x/u] & (\lambda x.t)\mathtt{L}_1[y/v]\mathtt{L}_2 u \to_{\mathtt{w}} (\lambda x.t)\mathtt{L}_1\mathtt{L}_2 u \\ \downarrow_{\mathtt{dB}} \quad \downarrow_{\mathtt{dB}} & \downarrow_{\mathtt{dB}} \quad \downarrow_{\mathtt{dB}} \\ \mathtt{C}[\![x]\!][x/u'] \to_{\mathtt{ls}} \mathtt{C}[\![u']\!][x/u'] & t[x/u]\mathtt{L}_1[y/v]\mathtt{L}_2 \to_{\mathtt{w}} t[x/u]\mathtt{L}_1\mathtt{L}_2 \end{array}$$

$$\begin{array}{c} (\lambda x.t)\mathtt{L}_{11}[z/\mathtt{C}[\![y]\!]]\mathtt{L}_{12}[y/v]\mathtt{L}_2 u \to_{\mathtt{ls}} (\lambda x.t)\mathtt{L}_{11}[z/\mathtt{C}[\![v]\!]]\mathtt{L}_{12}[y/v]\mathtt{L}_2 u \\ \downarrow_{\mathtt{dB}} \quad \downarrow_{\mathtt{dB}} \\ t[x/u]\mathtt{L}_{11}[z/\mathtt{C}[\![y]\!]]\mathtt{L}_{12}[y/v]\mathtt{L}_2 \to_{\mathtt{ls}} t[x/u]\mathtt{L}_{11}[z/\mathtt{C}[\![v]\!]]\mathtt{L}_{12}[y/v]\mathtt{L}_2 \end{array}$$

◀

The next goal is to establish confluence of the subsystem $\to_{\mathtt{lsub}}$. For that we first need to establish termination, that can be proved using some intermediate auxiliary results.

▶ **Lemma 6.** *Let $x \neq y$ and assume $y \notin \mathtt{fv}(v)$. Then $\mathtt{ml}_x(\mathtt{C}[\![y]\!]) + \mathtt{ml}_y(\mathtt{C}[\![y]\!]) \cdot \mathtt{ml}_x(v) = \mathtt{ml}_x(\mathtt{C}[\![v]\!]) + \mathtt{ml}_y(\mathtt{C}[\![v]\!]) \cdot \mathtt{ml}_x(v)$.*

Proof. By induction on $\mathtt{C}$. ◀

The following lemma states compatibility of $\to_{\mathtt{lsub}}$ w.r.t $\mathtt{ml}_x(\cdot)$.

▶ **Lemma 7** (Compatibility). *For all x, t and t', such that $t \to_{\mathtt{lsub}} t'$, $\mathtt{ml}_x(t) \geq \mathtt{ml}_x(t')$.*

Proof. The proof is by induction on the reduction relation. We show the base cases, since the inductive cases are straightfoward:
Case $\mathtt{C}[\![z]\!][z/u] \to_{\mathtt{ls}} \mathtt{C}[\![u]\!][z/u]$: $\mathtt{ml}_x(\mathtt{C}[\![z]\!][z/u]) \geq \mathtt{ml}_x(\mathtt{C}[\![u]\!][z/u])$ if and only if $\mathtt{ml}_x(\mathtt{C}[\![z]\!]) + \mathtt{ml}_z(\mathtt{C}[\![z]\!]) \cdot \mathtt{ml}_x(u) \geq \mathtt{ml}_x(\mathtt{C}[\![u]\!]) + \mathtt{ml}_z(\mathtt{C}[\![u]\!]) \cdot \mathtt{ml}_x(u)$, which holds by Lem. 6, since $z \notin \mathtt{fv}(u)$. Actually, in this case the equality holds.
Case $t[z/u] \to_{\mathtt{w}} t$: $\mathtt{ml}_x(t[z/u]) = \mathtt{ml}_x(t) + \mathtt{ml}_x(u) + \mathtt{ml}_z(t) \cdot \mathtt{ml}_x(u) \geq \mathtt{ml}_x(t)$. ◀

▶ **Lemma 8.** *Let $x \neq y$ such that $x, y \notin \mathtt{fv}(v)$. Then $\mathtt{ml}_x(C[\![x]\!]) > \mathtt{ml}_x(C[\![v]\!])$ and $\mathtt{ml}_y(C[\![x]\!]) = \mathtt{ml}_y(C[\![v]\!])$.*

Proof. The proof is by simultaneous induction on C. ◀

▶ **Lemma 9.** *The system $\to_{\mathtt{lsub}}$ is terminating on metaterms.*

Proof. We show that $t \to_{\mathtt{lsub}} t'$ implies $\mathtt{sz}(t) > \mathtt{sz}(t')$ so that $\to_{\mathtt{lsub}}$ is necessarily terminating. The proof is by induction on $t \to_{\mathtt{lsub}} t'$ and uses Lem. 7 and Lem. 8. ◀

▶ **Lemma 10.** *The reduction relations $\to_{\mathtt{dB}}$ and $\to_{\mathtt{lsub}}$ are confluent.*

Proof. As noticed before the relation $\to_{\mathtt{dB}}$ is trivially confluent. Since $\to_{\mathtt{lsub}}$ is terminating on metaterms (Lem. 9), in order to conclude confluence it is enough to verify joinability of all critical pairs. The sole critical peak is built overlapping the $\mathtt{ls}$-rule with itself: $\mathtt{C}[\![u]\!][x/u] \;{}_{\mathtt{ls}}\!\leftarrow \mathtt{C}[\![x]\!][x/u] = t[x/u] = \mathtt{C}'[\![x]\!][x/u] \to_{\mathtt{ls}} \mathtt{C}'[\![u]\!][x/u]$. In other words, t can be written as $\mathtt{D}'[\![x, x]\!]$, where $\mathtt{D}'$ is a two-hole context. Then, the critical peak is joinable: $\mathtt{D}'[\![u, x]\!][x/u] \to_{\mathtt{ls}} \mathtt{D}'[\![u, u]\!][x/u] \;{}_{\mathtt{ls}}\!\leftarrow \mathtt{D}'[\![x, u]\!][x/u]$. ◀

We can now conclude metaconfluence for the linear substitution calculus as follows:

▶ **Corollary 11.** *The reduction relation $\to_{\lambda_{\mathtt{lsub}}}$ is confluent on metaterms.*

Proof. Let $\to_{\lambda_{\mathtt{lsub}}} := \to_{\mathtt{dB}} \cup \to_{\mathtt{lsub}}$. Since both $\to_{\mathtt{dB}}$ and $\to_{\mathtt{lsub}}$ are confluent (Lem 10) and strongly commute (Theorem 5), their union is confluent using the Hindley-Rosen Theorem [10] introduced in Sec. 1. ◀

4.3 The Structural Lambda Calculus enjoys MetaConfluence

In this section we prove metaconfluence for the structural lambda calculus. As in the previous section, we first prove that the systems $\to_{\mathtt{dB}}$ and $\to_{\mathtt{str}}$ strongly commute, where, as defined in Sec. 3, we have $\to_{\mathtt{str}} := \to_{\mathtt{c}} \cup \to_{\mathtt{d}} \cup \to_{\mathtt{w}}$.

▶ **Theorem 12** (Strong Commutation). *$\forall t_0, t_1, t_2$, if $t_0 \to_{\mathtt{str}} t_1$ and $t_0 \to_{\mathtt{dB}} t_2$, $\exists t_3$ s.t. $t_1 \twoheadrightarrow_{\mathtt{dB}} t_3$ and $t_2 \overset{=}{\to}_{\mathtt{str}} t_3$.*

Proof. By induction on the reduction relations. We only show here the diagrams of the more interesting cases.

$$\begin{array}{ccc} (\lambda x.t)\mathrm{L}_1[y/v]\mathrm{L}_2u & \to_{\mathtt{w}} & (\lambda x.t)\mathrm{L}_1\mathrm{L}_2u \\ \downarrow_{\mathtt{dB}} & & \downarrow_{\mathtt{dB}} \\ t[x/u]\mathrm{L}_1[y/v]\mathrm{L}_2 & \to_{\mathtt{w}} & t[x/u]\mathrm{L}_1\mathrm{L}_2 \end{array} \qquad \begin{array}{ccc} (\lambda x.t)\mathrm{L}_1[y/v]\mathrm{L}_2u & \to_{\mathtt{d}} & (\lambda x.t)\mathrm{L}_1[\![y/v]\!]\mathrm{L}_2u \\ \downarrow_{\mathtt{dB}} & & \downarrow_{\mathtt{dB}} \\ t[x/u]\mathrm{L}_1[y/v]\mathrm{L}_2 & \to_{\mathtt{d}} & t[x/u]\mathrm{L}_1[\![y/v]\!]\mathrm{L}_2 \end{array}$$

$$\begin{array}{ccc} t[x/u] & \to_{\mathtt{c}} & t_{\langle y|x\rangle}[x/u][y/u] \\ \downarrow_{\mathtt{dB}} & & \downarrow_{\mathtt{dB}} \\ t[x/u'] & \to_{\mathtt{c}} & t_{\langle y|x\rangle}[x/u'][y/u'] \end{array} \qquad \begin{array}{ccc} t[x/u] & \to_{\mathtt{c}} & t_{\langle y|x\rangle}[x/u][y/u] \\ \downarrow_{\mathtt{dB}} & & \downarrow_{\mathtt{dB}} \\ t'[x/u] & \to_{\mathtt{c}} & t'_{\langle y|x\rangle}[x/u][y/u] \end{array}$$

$$\begin{array}{ccc} (\lambda x.t)\mathrm{L}_1[y/v]\mathrm{L}_2u & \to_{\mathtt{c}} & (\lambda x.t)\mathrm{L}_{1\,\langle y|y'\rangle}[y/v][y'/v]\mathrm{L}_2u \\ \downarrow_{\mathtt{dB}} & & \downarrow_{\mathtt{dB}} \\ t[x/u]\mathrm{L}_1[y/v]\mathrm{L}_2 & \to_{\mathtt{c}} & t[x/u]\mathrm{L}_{1\,\langle y|y'\rangle}[y/v][y'/v]\mathrm{L}_2 \end{array}$$

The subtle point here is that in the last two diagrams we need to choose the replacement $t'_{\langle y|x\rangle}$ (resp. $(t[x/u]\mathrm{L}_1)_{\langle y|y'\rangle}$) according to that we used for $t_{\langle y|x\rangle}$ (resp. $((\lambda x.t)\mathrm{L}_1)_{\langle y|y'\rangle}$). ◀

The next goal is to establish confluence of the subsystem $\to_{\mathtt{str}}$. For that we first need to establish termination, that can be proved using some intermediate auxiliary results.

▶ **Lemma 13.** *If $x \notin \mathtt{fv}(u)$ and $x \neq y$, then $\mathtt{P}_x(t) = \mathtt{P}_x(t[\![y/u]\!])$.*

Proof. The hypothesis implies $\mathtt{P}_x(u) = 0$. Moreover, we can easily prove by induction on t that $\mathtt{P}_x(t\{\!\{y/u\}\!\}) \stackrel{*}{=} \mathtt{P}_x(t)$, if $x \notin \mathtt{fv}(u)$ and $x \neq y$. We then consider all the possible cases for $t[\![y/u]\!]$.

1. If $y \in \mathtt{fm}(t)$ and $y \in \mathtt{fr}(t)$, then $\mathtt{P}_x(t[\![y/u]\!]) = \mathtt{P}_x(t\{\!\{y/u\}\!\}[y/u]) = \mathtt{P}_x(t\{\!\{y/u\}\!\}) \stackrel{*}{=} \mathtt{P}_x(t)$;
2. If $y \notin \mathtt{fm}(t)$ and $y \in \mathtt{fr}(t)$, then $\mathtt{P}_x(t[\![y/u]\!]) = \mathtt{P}_x(t\{\!\{y/u\}\!\}) \stackrel{*}{=} \mathtt{P}_x(t)$;
3. If $y \in \mathtt{fm}(t)$ and $y \notin \mathtt{fr}(t)$, then $\mathtt{P}_x(t[\![y/u]\!]) = \mathtt{P}_x(t[y/u]) = \mathtt{P}_x(t\{\!\{y/u\}\!\}) \stackrel{*}{=} \mathtt{P}_x(t)$;
4. If $y \notin \mathtt{fv}(t)$, then $\mathtt{P}_x(t[\![y/u]\!]) = \mathtt{P}_x(t)$. ◀

The following properties hold for metaterms. The proofs can be done by simple structural induction, where the metavariable case is straightforward and the other cases are in [6].

▶ **Lemma 14.** *Let t be a metaterm. Then*

1. *$|t|_x \leq \mathtt{P}_x(t)$.*
2. *If $x \notin \mathtt{fv}(u)$ then $\mathtt{P}_x(t) = \mathtt{P}_x(t[y/u])$.*
3. *If x, y, z are pairwise distinct and $z \notin \mathtt{fv}(t)$ then $\mathtt{P}_x(t) = \mathtt{P}_x(t_{\langle y|z\rangle})$.*
4. *If $y \notin \mathtt{fv}(t)$ and $t' = t_{\langle x|y\rangle}$ then $\mathtt{P}_x(t) = \mathtt{P}_x(t') + \mathtt{P}_y(t')$.*

▶ **Lemma 15.** *If $|t|_y = 1$ then $\mathtt{P}_x(t\{\!\{y/u\}\!\}) \leq \mathtt{P}_x(t) + \mathtt{P}_y(t) \cdot \mathtt{P}_x(u)$.*

Proof. By induction on t. ◀

▶ **Lemma 16.** *If $|t|_x = 1$ and $x \notin \mathtt{fm}(t)$ then $\mathtt{jm}(t[x/u]) \sqsupset \mathtt{jm}(t[\![x/u]\!])$.*

Proof. By induction on t using Lem. 13 and Lem. 14. ◀

▶ **Lemma 17.** *Let t, t' be metaterms. If $t \to_{\mathtt{str}} t'$ then $\mathtt{P}_x(t) \geq \mathtt{P}_x(t')$.*

Proof. By induction on $t \to_{\mathtt{str}} t'$ using Lem. 15 and Lem. 14. ◀

▶ **Lemma 18.** *The reduction relation $\to_{\mathtt{str}}$ is terminating on metaterms.*

Proof. By induction on $t \to_{\mathtt{str}} t'$ using Lem. 16 and Lem. 17. ◀

▶ **Lemma 19.** *The reduction relations* $\to_{\mathtt{str}}$ *and* $\to_{\mathtt{dB}}$ *are confluent.*

Proof. The reduction relation $\to_{\mathtt{str}}$ is terminating on metaterms (Lem. 18), and do not have critical pairs because its rules are mutually exclusive. Therefore, $\to_{\mathtt{str}}$ is confluent. ◀

Metaconfluence of the structural substitution calculus is then obtained as follows:

▶ **Corollary 20.** *The reduction relation* $\to_{\lambda_{\mathtt{str}}}$ *is confluent on metaterms.*

Proof. Let $\to_{\lambda_{\mathtt{str}}} := \to_{\mathtt{dB}} \cup \to_{\mathtt{str}}$. Both $\to_{\mathtt{dB}}$ and $\to_{\mathtt{str}}$ are confluent (Lem. 19) and strongly commute (Theorem 12), therefore their union is confluent by the Hindley-Rosen Theorem [10] introduced in Sec. 1. ◀

5 Conclusion

We define reduction for metaterms for three calculi with explicit substitutions that act at a distance, namely the substitution calculus, the linear substitution calculus and the structural lambda calculus. This is done by defining a subtle notion of metasubstitution, which is completely fixed for metavariables. In contrast to other specifications of λ-calculi with ES on metaterms, our resulting reduction systems do not contain equations, so that their equational theories are simple enough to be treated with simple rewriting techniques. In particular, our proofs of (meta)confluence can be achieved by using the well-known Hindley-Rosen Theorem.

As mentioned before, metaconfluence is an essential property of calculi with ES used to implement higher-order unification (HOU) procedures [15, 8]. Indeed, such algorithms need to compare typed metaterms in (η-long) normal form, which are unique by metaconfluence. They then generate new metavariables in order to denote partial solutions that need again to be in (η-long) normal form in order to recursively apply the algorithm. As future work, we want to investigate higher-order unification (HOU) procedures based on calculi acting at a distance. We believe that the simplicity and applicability of such calculi can lead to unification procedures that are simpler than already known unification procedures based on other ES calculi [15, 8].

References

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

2 B. Accattoli, P. Barenbaum, and D. Mazza. Distilling abstract machines. Available at `https://sites.google.com/site/beniaminoaccattoli/`, 2014.

3 B. Accattoli, E. Bonelli, D. Kesner, and C. Lombardi. A nonstandard standardization theorem. In 41st Symposium on Principles of Programming Languages (POPL), editor, *ACM SIGPLAN-SIGACT*, pages 659–670, 2014.

4 B. Accattoli and D. Kesner. The structural lambda-calculus. In *19th EACSL Annual Conference on Computer Science and Logic (CSL)*, volume 6247 of *LNCS*, pages 381–395. Springer-Verlag, 2010.

5 B. Accattoli and D. Kesner. The permutative lambda calculus. In *LPAR*, pages 23–36, 2012.

6 B. Accattoli and D. Kesner. Preservation of strong normalisation modulo permutations for the structural lambda-calculus. *Logical Methods in Computer Science*, 8(1), 2012.

7 B. Accattoli and U. Dal Lago. Beta reduction is invariant, indeed. Accepted to LICS/CSL 2014.

8 M. Ayala-Rincón and F. Kamareddine. Unification via the λs_e-Style of Explicit Substitution. *The Logical Journal of the IGPL*, 9(4):489–523, 2001.

9 F. Baader and T. Nipkow. *Term Rewriting and* All That. Cambridge University Press, 1998.

10 H. Barendregt. *The Lambda Calculus – Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1984.

11 M. Bezem, J. W. Klop, and R. de Vrijer, editors. *Term Rewriting Seminar – Terese*. Cambridge University Press, 2003.

12 P.-L. Curien, T. Hardin, and J.-J. Levy. Confluence properties of weak and strong calculi of explicit substitutions. *Journal of the ACM*, 43:43–2, 1996.

13 R. David and B. Guillaume. A lambda-calculus with explicit weakening and explicit substitution. *Mathematical Structures in Computer Science*, 11(1):169–206, 2001.

14 F. L. C. de Moura, M. Ayala-Rincón, and F. Kamareddine. Higher-Order Unification: A structural relation between Huet's method and the one based on explicit substitutions. *Journal of Applied Logic*, 6(1):72–108, 2008.

15 G. Dowek, T. Hardin, and C. Kirchner. Higher order unification via explicit substitutions. *Inf. Comput.*, 157(1-2):183–235, 2000.

16 G. Dowek, T. Hardin, C. Kirchner, and F. Pfenning. Unification via explicit substitutions: The case of higher-order patterns. In *JICSLP*, pages 259–273, 1996.

17 Z. el A. Benaissa, D. Briaud, P. Lescanne, and J. Rouyer-Degli. $\lambda\upsilon$, a Calculus of Explicit Substitutions which Preserves Strong Normalization. *JFP*, 6(5):699–722, 1996.

18 B. Guillaume. The λs_e-calculus Does Not Preserve Strong Normalization. *J. of Func. Programming*, 10(4):321–325, 2000.

19 F. Kamareddine and A. Ríos. A λ-calculus à la de Bruijn with Explicit Substitutions. In *Proc. of PLILP'95*, volume 982 of *LNCS*, pages 45–62. Springer, 1995.

20 F. Kamareddine and A. Ríos. Extending a λ-calculus with Explicit Substitution which Preserves Strong Normalisation into a Confluent Calculus on Open Terms. *Journal of Functional Programming*, 7:395–420, 1997.

21 D. Kesner. The theory of calculi with explicit substitutions revisited. In *CSL*, pages 238–252, 2007.

22 D. Kesner. Perpetuality for full and safe composition (in a constructive setting). In *To appear in Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP 2008), Track B*, Reykjavik, Iceland,, 2008.

23 D. Kesner. A Theory of Explicit Substitutions with Safe and Full Composition. *Logical Methods in Computer Science*, 5(3:1):1–29, 2009.

24 D. Kesner and S. Ó Conchúir. Milner's Lambda Calculus with Partial Substitutions. Technical Report, Université Paris Diderot, 2008.

25 P.-A. Melliès. Typed λ-calculi with explicit substitutions may not terminate. In *Proceedings of TLCA'95*, volume 902 of *LNCS*. Springer-Verlag, 1995.

26 R. Milner. Local bigraphs and confluence: Two conjectures: (extended abstract). *ENTCS*, 175(3):65–73, 2007.

27 F. Renaud. *Les Ressources Explicites vues par la Théorie de la Réécriture*. PhD thesis, Université Paris Diderot - Paris 7, 2011. Available at `www.lix.polytechnique.fr/~renaud/these.pdf`.

28 Jean yves Girard. Proof-nets: The parallel syntax for proof-theory. In *Logic and Algebra*, pages 97–124. Marcel Dekker, 1996.