Expression Reduction Systems with Patterns

Julien Forest[1] and Delia Kesner[2]

[1] LRI (CNRS UMR 8623), Université Paris-Sud, France, `forest@lri.fr`.
[2] PPS (CNRS UMR 7126), Université Paris 7, France, `kesner@pps.jussieu.fr`.

Abstract. We introduce a new higher-order rewriting formalism, called *Expression Reduction Systems with Patterns (ERSP)*, where abstraction is not only allowed on variables but also on nested patterns. These patterns are built by combining standard *algebraic* patterns with *choice* constructors used to denote different possible structures allowed for an abstracted argument. In other words, the non deterministic choice between different rewriting rules which is inherent to classical rewriting formalisms can be lifted here to the level of patterns. We show that confluence holds for a reasonable class of systems and terms.

Introduction

Higher-order rewrite systems are able to combine formalisms coming from proof theory, such as λ-calculus, with formalisms arising in algebraic specifications, such as first-order rewrite systems [17, 20, 26]. The main idea behind higher-order rewriting concerns the transformation of terms in the presence of *binding* mechanisms for variables and substitutions. Thus for example, functional and logic programming, equational reasoning, object-oriented programming, concurrent systems and theorem provers may be all be encoded by higher-order rewrite systems.

Many higher-order rewrite systems exist in the literature starting at the seminal work by J-W. Klop [25]. Many other interesting formalisms [5, 24, 30, 35, 38] were introduced later. The theory of higher-order rewriting is considerably more involved than that of first-order rewriting; many articles were consecrated to the study of its foundations, applications, semantics and implementation.

In all the higher-order formalisms mentioned before the binding mechanism is only allowed on *variables*. However, most popular functional languages [6, 7, 19, 29, 34] and proof assistants [1, 12, 13, 21, 28, 32] allow definitions by cases via pattern-matching mechanisms. Thus, a natural extension of higher-order rewriting consists in the introduction of binders for patterns so that a projection function like $\lambda\langle x, y\rangle.x$ would be also acceptable.

The *Pattern-Matching Calculus* [23], proposed as a theoretical framework to study pattern-matching in a *pure functional paradigm*, allows precisely this kind of binding mechanisms. Its evaluation process is given by the following generalization of the standard β-rule to the case of patterns:

$$(\beta_{PM}) \qquad app(\lambda\mathbb{X}.M, N) \longrightarrow M\{\mathbb{X} \texttt{ by } N\}$$

where $\mathbb{X}$ denotes a pattern and $\{\mathbb{X} \texttt{ by } N\}$ denotes a substitution resulting from the pattern-matching operation on the pattern $\mathbb{X}$ and the term N.

This calculus was later extended with *explicit operators* [9, 8, 14]; weak reduction was widely studied in [14]. Another language allowing abstractions on patterns is the ρ-calculus [11], for which typing [18], explicit operators [10] and exception mechanisms [16] were defined and analyzed.

In this paper we introduce a new higher-order formalism, called *Expression Reduction Systems with Patterns* (ERSP), where binding mechanisms are allowed on complex patterns. Our calculus constitutes an extension of ERS [24] and SERS [5] to the case of patterns, and a generalization of the Pattern-Matching Calculus to the case of general higher-order rewriting (and not only functional rewriting). ERSP patterns are defined as combinations of standard *algebraic* structures with special *choice* constructors used to denote different possible syntactic forms for any abstracted argument. Thus for example, the function which computes the length of a given list may be specified as the following ERSP term:

$$\lambda\, a\langle nil, cons(x,t)\rangle . a\langle 0, 1 + len(t)\rangle$$

where λ is the classical function binder, a is a special variable used to identify the set of *choice* patterns nil and $cons(x,t)$ with the set of their corresponding continuations 0 and $1 + len(t)$.

We carefully extend all the expected notions of rewriting to our framework, namely, terms, metaterms, rewrite rules, substitutions, reduction, etc. We then identify a class of ERSP, called *orthogonal l-constructor* systems, and a class of terms, called *l-constructor deterministic terms*, for which confluence holds. More precisely, reduction on this class of terms via this class of systems corresponds to reduction on ordinary terms (without patterns) in classical orthogonal higher-order systems [30, 36]. Much more, our confluence result turns out to give in particular a confluence result for SERS.

The paper is organized as follows. Section 1 introduces the basic ingredients which are needed to define ERSP systems while Section 2 defines ERSP rewrite systems and their basic associated reduction relations. In Section 3 we develop a concrete example of reduction in our framework. Section 4 is consecrated to restrict the class of ERSP and the notion of reduction so that confluence of this restricted reduction relation on this restricted class of systems will follow (Section 5). We conclude and give many further research directions in Section 6. By lack of space we cannot provide here all the proofs but a complete version of this work can be found as [15].

1 Basic notions of the ERSP formalism

We consider the following denumerable and disjoint infinite sets:

- A set $\mathcal{UV}$ of *usual variables* denoted $x, y, z, \ldots$
- A set $\mathcal{CV}$ of *choice variables* denoted $a, b, c, \ldots$
- A set $\mathcal{PV}$ of *pattern metavariables* denoted $\mathbb{X}, \mathbb{Y}, \ldots$
- A set $\mathcal{TV}$ of *term metavariables* denoted $M, N, \ldots$
- A set $\mathcal{F}$ of *function symbols* equipped with a fixed (possibly zero) arity, denoted $f, g, h, \ldots$
- A set $\mathcal{B}$ of *binder symbols* denoted $\lambda, \mu, \nu, \ldots$

When no special distinction is needed for the previous sets of variables and metavariables will use the symbols $\widehat{x}, \widehat{y}, \widehat{z}, \ldots$.

Metapatterns (p) and metaterms (t) are generated by the grammars:

$p ::=$	x	usual variable	$t ::=$	x	usual variable
$\mid$	$\mathbb{X}$	pattern metavariable	$\mid$	M	term metavariable
$\mid$	$f(p,\ldots,p)$	algebraic	$\mid$	$f(t,\ldots,t)$	algebraic
$\mid$	$a\langle p,\ldots,p\rangle$	choice	$\mid$	$a\langle t,\ldots,t\rangle$	case
$\mid$	$@(p,\ldots,p)$	contraction	$\mid$	$\mu p.t$	abstraction
$\mid$	$\_$	wildcard	$\mid$	$t\{p \texttt{ by } t\}$	pattern-matching

The pattern constructor @() is varyadic, i.e. it has no fix arity. As we will see later, the wildcard pattern _ can be considered as a special case of @() applied to 0 arguments. The constructor $a\langle\ \rangle$ is also varyadic, but with an arity different from 0. We assume that whenever a choice variable a appears inside t, then all its occurrences have the same arity. As a consequence, a term like $\mu a\langle x\rangle.a\langle x,y\rangle$ is not allowed. The symbol $\{\ \mathtt{by}\ \}$ is called the pattern-matching constructor. The metaterms $\mu p.t$ and $t\{p\ \mathtt{by}\ t'\}$ define bindings whose scope is t for all the (usual and choice) variables occurring in p.

A metapattern (resp. metaterm) is said to be a *pattern* (resp. *preterm*) if it contains no metavariables. A preterm is said to be a *term* if it contains no pattern-matching constructors.

We denote by $\mathcal{MV}(p)$ (resp. $\mathcal{V}ar(p)$) the set of all the *pattern* metavariables (resp. variables) appearing in a metapattern (resp. pattern) p. We denote by $\mathcal{MV}(t)$ the set of all the *term* metavariables appearing in t.

Definition 1. *A metapattern is called linear if each variable and metavariable appears at most once in it. We use the notation $p \in p'$ to say that the metapattern p appears inside the metapattern p'. A metaterm t is called p-linear iff every metapattern p in t is linear.*

Let us illustrate the use of our syntax by considering the `fibonacci` function specified by the equations `fib(0)=0, fib(1)=1` and `fib(x+2)=fib(x)+fib(x+1)`.

Using a choice variable a of arity 3 to encode the three different choices given by the previous specification, one possible specification of `fib` in our syntax is:

$$fib(M) \longrightarrow\ app(\lambda\ a\langle 0, s(0), s(s(x))\rangle.a\langle 0, s(0), app(fib, x) + app(fib, s(x))\rangle, M)$$

where app is the application symbol, λ is the classical function binder, and natural numbers are encoded by 0, $s(0)$, $s(s(0))$, ...

Definition 2 (Positions of metaterms). *A position is a word over the alphabet $\mathbb{N}$; we use ϵ to denote the empty word. The set of positions of a metaterm t, denoted $\mathcal{POS}(t)$, is defined by induction as follows:*

- $\epsilon \in \mathcal{POS}(t)$
- $i.q \in \mathcal{POS}(f(s_1,\ldots,s_m))$ *if* $i \in \{1,\ldots,m\}$ *and* $q \in \mathcal{POS}(s_i)$
- $1.q \in \mathcal{POS}(\mu p.s)$ *if* $q \in \mathcal{POS}(s)$
- $i.q \in a\langle s_1,\ldots,s_m\rangle$ *if* $i \in \{1,\ldots,m\}$ *and* $q \in \mathcal{POS}(s_i)$
- $1.1.q \in \mathcal{POS}(s\{p\ \mathtt{by}\ u\})$ *if* $q \in \mathcal{POS}(s)$
- $2.q \in \mathcal{POS}(s\{q\ \mathtt{by}\ u\})$ *if* $q \in \mathcal{POS}(u)$

Definition 3 (Submetaterms). *Given a metaterm s and $p \in \mathcal{POS}(s)$, we define the submetaterm of s at position q, written $s|_q$, as a metaterm appearing inside s defined as follows:*

$$\begin{array}{ll}
s|_\epsilon & = s \\
f(s_1,\ldots,s_n)|_{i.q} & = s_i|_q \\
a\langle s_1,\ldots,s_n\rangle|_{i.q} & = s_i|_q \\
\mu p.s|_{1.q} & = s|_q \\
u\{p\ \mathtt{by}\ v\}|_{1.1.q} & = u|_q \\
u\{p\ \mathtt{by}\ v\}|_{2.q} & = v|_q
\end{array}$$

When $t|_p = q$, we will say that p is an *occurrence* of q in t.

The following notion is used to describe the set of variables/metavariables appearing along a given path which will play latter a role of "bound" objects in a terms/metaterms (this concept is formally introduced in Definition 5).

Definition 4 (Parameter Path). *Given a metaterm s and $p \in \mathcal{POS}(s)$, we define the parameter path of s at position q, written $\mathcal{PP}(s,q)$, as the following subset of variables and metavariables of s:*

$$\begin{array}{ll}
\mathcal{PP}(s,\epsilon) & = \emptyset \\
\mathcal{PP}(f(s_1,\ldots,s_n), i.q) & = \mathcal{PP}(s_i,q), \ \textit{for } i \in \{1\ldots n\} \\
\mathcal{PP}(a\langle s_1,\ldots,s_n\rangle, i.q) & = \mathcal{PP}(s_i,q), \ \textit{for } i \in \{1\ldots n\} \\
\mathcal{PP}(\mu p.s, 1.q) & = \mathcal{V}ar(p) \cup \mathcal{MV}(p) \cup \mathcal{PP}(s,q) \\
\mathcal{PP}(u\{p \texttt{ by } v\}, 1.1.q) & = \mathcal{V}ar(p) \cup \mathcal{MV}(p) \cup \mathcal{PP}(u,q) \\
\mathcal{PP}(u\{p \texttt{ by } v\}, 2.q) & = \mathcal{PP}(v,q)
\end{array}$$

As an example, if $t = M\{g(\mathbb{X},x) \texttt{ by } \mu a\langle \mathbb{Y}, s(\mathbb{Y})\rangle.N\}$, then we have $t|_2 = \mu a\langle \mathbb{Y}, s(\mathbb{Y})\rangle.N$, $\mathcal{PP}(t,2) = \emptyset$, $t|_{1.1} = M$, $\mathcal{PP}(t,1.1) = \{\mathbb{X}, x\}$, $t|_{2.1} = N$ and $\mathcal{PP}(t,2.1) = \{\mathbb{Y}, a\}$.

We assume that different "non parallel" metapatterns appearing on a same path cannot share (meta)variables. Thus for example, we will not consider metaterms like $\mu\mathbb{X}.\lambda\mathbb{X}.M$ or $\lambda x.\mu x.M$ but the metaterm t given below is allowed. This is just a generalization of Barendregt's convention on bound variables in the framework of the λ-calculus.

Definition 5 (Free Variables). *The set of free (meta)variables of a metaterm is defined by induction as follows:*

$$\begin{array}{ll}
\mathcal{FV}(x) & = \{x\} \\
\mathcal{FV}(M) & = \{M\} \\
\mathcal{FV}(f(t_1,\ldots,t_n)) & = \mathcal{FV}(t_1) \cup \ldots \mathcal{FV}(t_n) \\
\mathcal{FV}(a\langle t_1,\ldots,t_n\rangle) & = \mathcal{FV}(t_1) \cup \ldots \mathcal{FV}(t_n) \cup \{a\} \\
\mathcal{FV}(\mu p.u) & = \mathcal{FV}(u) \setminus \mathcal{V}ar(p) \\
\mathcal{FV}(t\{p \texttt{ by } u\}) & = (\mathcal{FV}(t) \setminus \mathcal{V}ar(p)) \cup \mathcal{FV}(u)
\end{array}$$

All the variables appearing in a metaterm t that are not free are called *bound variables*. Without loss of generality we assume the sets of free and bound variables to be disjoint. We work modulo α-conversion on preterms, so that renaming of bound variables is used when necessary to avoid clashes. Thus for example $\mu a\langle x,y,z\rangle.a\langle x,x,v\rangle =_\alpha \mu b\langle x',y',z'\rangle.b\langle x',x',v\rangle$.

Definition 6 (Well-formed metaterm). *A metaterm t is well-formed iff t has no free occurrences of choice/usual variables.*

The metaterms $\mu x.M$, $\mu\mathbb{X}.M$, $\mu x.f(M,x)$ and $\mu a\langle x,y\rangle.a\langle x,y\rangle$ are well-formed while $f(a\langle g,g\rangle)$ and $f(x)$ are not.

Definition 7. *The set of rigid positions of t, denoted $\mathcal{RP}(t)$, is given by:*

$$\{q \in \mathcal{POS}(t) \mid t|_q \textit{ is not a term variable}\}$$

The following notion is used to denote the free variables of a term which remain *after* a given choice on a choice variable.

Definition 8 (Localized Free Variables). *Given $a \in \mathcal{CV}$, $i \geq 1$ and a preterm t, we define the set $\mathcal{FV}^i_a(t)$ of* localized free variable *of t exactly as done for the set of free variables (Definition 5) except for the following case:*

$$\mathcal{FV}^i_a(a\langle t_1,\ldots,t_n\rangle) = \mathcal{FV}^i_a(t_i) \ \textit{if } 1 \leq i \leq n$$

Indeed, $\mathcal{FV}^i_a(b\langle x,y,z\rangle) = \{b,z,x,y\}$ for any i and $\mathcal{FV}^1_a(a\langle x,y,z\rangle) = \{x\}$. Moreover, as we work modulo α-conversion we have $\mathcal{FV}^1_a(\mu a\langle x,y\rangle.a\langle f(x,z),u\rangle) = \mathcal{FV}^1_a(\mu b\langle x,y\rangle.b\langle f(x,z),u\rangle) = \{z,u\}$.

Remark 1. We remark that when the choice variable a is not free in t then we have $\mathcal{FV}^i_a(t) = \mathcal{FV}(t)$ for every i.

Definition 9 (Acceptable preterms).

Acceptability is the least relation on preterms containing the variables such that:

- *If $t_1, \ldots, t_n$ are acceptable, then $f(t_1, \ldots, t_n)$ and $a\langle t_1, \ldots, t_n\rangle$ are acceptable for any $f \in \mathcal{F}$ and any $a \in \mathcal{CV}$.*
- *If t is acceptable and p is a pattern, then for all $a\langle p_1, \ldots, p_n\rangle \in p$, for all $i \in 1 \ldots n$, and for all $j \neq i$ such that $(\mathcal{FV}^j_a(t) \setminus \mathcal{V}ar(p_j)) \cap \mathcal{V}ar(p_i) = \emptyset$, we have that $\mu p.t$ is an acceptable term.*
- *If $\mu p.t$ and u are acceptable, then $t\{p \texttt{ by } u\}$ is acceptable.*

The role of acceptability is to prevent the creation of new free variables during the substitution process. Indeed, the terms $\mu a\langle x, x\rangle.a\langle x, x\rangle$ and $\mu a\langle x, y\rangle.a\langle x, y\rangle$ are acceptable while $\mu a\langle x, y\rangle.b\langle x, y\rangle$ is not since $\mathcal{FV}^1_a(b\langle x, y\rangle) \setminus \mathcal{V}ar(x) = \{y, b\}$ and $\{y, b\} \cap \mathcal{V}ar(y) = \{y\}$. The term $\mu a\langle x, y\rangle.a\langle y, x\rangle$ is neither acceptable: if we choose the first pattern branch x in the pattern $a\langle x, y\rangle$ we have then to consequently choose the first term branch y in the term $a\langle y, x\rangle$, thus y becomes a *new* free variable.

Precontexts are preterms with one (and only one) occurrence containing a distinguished constant called a "hole" (and denoted $\Box$). A *context* is a precontext with no occurrence of the pattern-matching constructor. We remark that the notion of acceptability is not closed by precontexts as for example the preterm $a\langle x, y\rangle$ is acceptable but $\lambda a\langle y, x\rangle.a\langle x, y\rangle$ is not.

Definition 10 (Metasubstitutions/Substitutions). *We define a metasubstitution θ as a denumerable set of pairs having either the form $x \triangleright t$, $a \triangleright i$, $\mathbb{X} \triangleright p$, or $M \triangleright t$, where t is a term[1], i is a natural number and p is a pattern. As a consequence each θ can be written as (θ_m, θ_v), where θ_m contains only the pairs $\mathbb{X} \triangleright p$ and $M \triangleright t$, and θ_v contains all the other ones. The application of θ to a (meta)variable $\widehat{x}$ is defined as $\theta\widehat{x} = o$ if $\widehat{x} \triangleright o \in \theta$, and $\theta\widehat{x} = \widehat{x}$, otherwise. We identify id with the empty substitution, i.e. $id\widehat{x} = \widehat{x}$ for every $\widehat{x}$. The domain of θ is given by $Dom(\theta) = \{\widehat{x} \mid \widehat{x} \triangleright o \in \theta \text{ and } o \neq \widehat{x}\}$ and the codomain of θ is given by $Codom(\theta) = \bigcup_{x \in Dom(\theta)} \mathcal{FV}(\theta x)$. A substitution θ is a metasubstitution such that $Dom(\theta_m) = \emptyset$.*

The following notion will play an important role in the definition of reduction relation (Definition 18).

Definition 11 (Well-formed metasubstitution). *A metasubstitution $\theta = (\theta_m, \theta_v)$ is said to be well-formed iff $(\bigcup_{M \in Dom(\theta_m)} \mathcal{FV}(\theta_m M)) \cap Dom(\theta_v) = \emptyset$.*

The *union* of two well-formed metasubstitutions θ_1 and θ_2 is denoted by $\theta_1 \sqcup \theta_2$. This union is *only defined* if the resulting metasubstitution is well-formed and if for every (meta)variable $\widehat{x} \in Dom(\theta_1) \cap Dom(\theta_2)$ we have $\theta_1\widehat{x} = \theta_2\widehat{x}$.

We are now ready to define the notion of *pattern-matching*. This operation is not defined in general as a function from patterns and terms to substitutions but from patterns and terms to *sets* of substitutions. We will see latter how to ensure the uniqueness of this result.

[1] and not a preterm

Definition 12 (Pattern-matching). *For each pair (p,t), where p is a pattern and t is a term, we associate a set of substitutions as follows:*

$$
\begin{array}{lll}
id & \in \{\!\{ \_ \;\texttt{by}\; t\}\!\} & \\
\{x \rhd t\} & \in \{\!\{x \;\texttt{by}\; t\}\!\} & \\
\theta_1 \sqcup \ldots \sqcup \theta_n & \in \{\!\{@(p_1,\ldots,p_n) \;\texttt{by}\; t\}\!\} & \textit{if } \theta_i \in \{\!\{p_i \;\texttt{by}\; t\}\!\} \\
\theta_1 \sqcup \ldots \sqcup \theta_n & \in \{\!\{f(p_1 \ldots p_n) \;\texttt{by}\; f(t_1 \ldots t_n)\}\!\} & \textit{if } \theta_i \in \{\!\{p_i \;\texttt{by}\; t_i\}\!\} \\
\{a \rhd i\} \sqcup \theta_i & \in \{\!\{a\langle p_1 \ldots p_n\rangle \;\texttt{by}\; t\}\!\} & \textit{if } \theta_i \in \{\!\{p_i \;\texttt{by}\; t\}\!\}
\end{array}
$$

We remark that in the last three cases the result of $\{\!\{p \;\texttt{by}\; t\}\!\}$ is defined only if $\sqcup$ is defined. We also remark that all the substitutions in $\{\!\{p \;\texttt{by}\; t\}\!\}$ are well-formed since they do not map metavariables. When $\{\!\{p \;\texttt{by}\; t\}\!\}$ is a singleton we will make an abuse of notation by writing $\{\!\{p \;\texttt{by}\; t\}\!\}$ to denote the only element of this set.

As an example of the previous definition, the pattern-matching $\{\!\{a\langle 0,x\rangle \;\texttt{by}\; 0\}\!\}$ has two solutions: $\{a \rhd 1\}$ and $\{a \rhd 2, x \rhd 0\}$. This comes from the fact that the pattern $a\langle 0,x\rangle$ contains two "overlapping" subpatterns 0 and x.

Definition 13 (Acceptable/linear metasubstitution). *Let S be a set of variables. A metasubstitution θ is said to be linear w.r.t. S iff for every (meta)variable $\widehat{x} \in S$, $\theta\widehat{x}$ is linear. A metasubstitution θ is said to be acceptable w.r.t. S iff for every (meta)variable $\widehat{x} \in S$, $\theta\widehat{x}$ is an acceptable term.*

A metasubstitution θ is said to be acceptable (resp. linear) if it is acceptable (resp. linear) w.r.t $Dom(\theta)$.

Acceptability enjoys the following property:

Lemma 1. *If t is an acceptable term and $\theta \in \{\!\{p \;\texttt{by}\; t\}\!\}$, then θ is an acceptable substitution.*

Proof. By induction on p.

- If $p = \_$ then $\theta = id$ and the result obviously holds.
- If $p = x$ then the result holds by hypothesis.
- If $p = f(p_1,\ldots,p_n)$ then by definition of $\{\!\{ \;\texttt{by}\; \}\!\}$ we know that $t = f(t_1,\ldots,t_n)$ and all the $\{\!\{p_i \;\texttt{by}\; t_i\}\!\}$ are defined. Since all the t_i are acceptable, thus by i.h.all the $\theta_i \in \{\!\{p_i \;\texttt{by}\; t_i\}\!\}$ are acceptable substitutions. Hence the result obviously holds.
- If $p = @(p_1,\ldots,p_n)$ or $p = a\langle p_1,\ldots,p_n\rangle$ we proceed as before.

It is time to make the point w.r.t capture of variables in higher-order rewriting.

In CRS [25, 27] for example, a metaterm like $\lambda x.M(x)$ allows the (eventual) capture of the variable x while $\lambda x.M$ does not. In this formalism the β-rule has to be written as $app(\lambda x.M(x), N) \longrightarrow M(N)$ which does not correspond to the traditional way to express the β-rule.

In ERS [24] there is a metasubstitution operator which allows to express the β-rule in a more traditional way as $app(\lambda x.M, N) \longrightarrow M\{x/N\}$. The instantiation of the metavariable M may or may not capture the variable x. However, we cannot assume α-conversion on metaterms in this formalism: if we suppose $\lambda x.M =_\alpha \lambda y.M$, then the instantiation of M by x will give two non α-equivalent terms $\lambda x.x \neq_\alpha \lambda y.x$. In order to properly handle α-conversion on terms but not on metaterms two *different* levels of syntax are needed, and this is the approach taken in general in the ERS formalism (see also [5]).

To allow α-conversion on the level of terms but not on that of metaterms a special notion of instantiation is needed, where the application of a metasubstitution $\theta = (\theta_m, \theta_v)$ to a metaterm will be split into two different steps: θ_m is used as *first-order* replacement, so that capture of variables can be provoked, while θ_v is used as *higher-order* substitution, so that no capture of variables is possible.

Definition 14 (Applying a metasubstitution). *Given a metasubstitution $\theta = (\theta_m, \theta_v)$ and a metaterm t, the application of θ to t (or instantiation of t by θ) yields a set of terms, written $\theta(t)$, which is computed in two steps:*

1. *First compute the first-order replacement $\theta_m(t)$ obtaining a preterm s (in the case where $\theta_m(t)$ is not still a preterm the application is not defined)*
2. *Then compute the set of terms $\theta_v(s)$, where θ_v is a higher-order substitution which works modulo α-conversion defined as follows:*

$$
\begin{array}{lll}
\theta_v x & \in \theta_v(x) & \textit{if } x \in Dom(\theta_v) \\
x & \in \theta_v(x) & \textit{if } x \notin Dom(\theta_v) \\
\mu p.t' & \in \theta_v(\mu p.t) & \textit{if } t' \in \theta_v(t) \textit{ and no capture of variables holds} \\
f(t'_1, \ldots, t'_n) & \in \theta_v(f(t_1, \ldots, t_n)) & \textit{if } t'_i \in \theta_v(t_i) \\
t'_i & \in \theta_v(a\langle t_1, \ldots, t_n\rangle) & \textit{if } \theta_v a = i \textit{ and } t'_i \in \theta_v(t_i) \\
a\langle t'_1, \ldots, t'_n\rangle & \in \theta_v(a\langle t_1, \ldots, t_n\rangle) & \textit{if } t'_i \in \theta_v(t_i) \textit{ and } a \notin Dom(\theta_v) \\
t' & \in \theta_v(t\{p \texttt{ by } u\}) & \textit{if } u' \in \theta_v(u), \theta'_v \in \{\!\{p \texttt{ by } u'\}\!\}, \\
 & & \; t' \in (\theta'_v \sqcup \theta_v)(t) \\
 & & \;\; \textit{and no capture of variables holds}
\end{array}
$$

When the metasubstitution θ is a substitution, we may make an abuse of notation by writing $\theta(t)$ instead of $\theta_v(t)$. We also remark that if a metaterm t has no pattern-matching constructor, then, if defined, $\theta(t)$ is a singleton.

Another interesting observation is that even when θ_v of some metasubstitution θ is empty, the second step of the previous procedure must be computed on preterms (which still have pattern-matching constructors to be eliminated).

Let us see how the application of a metasubstitution works on an example. Consider $\theta = (\theta_m, \theta_v)$, where $\theta_m = \{\mathbb{X}/a\langle x, f(z,y)\rangle, M/a\langle g(x,x), z\rangle, N/f(x,x)\}$ and $\theta_v = \emptyset$. In order to compute $\theta(M\{\mathbb{X} \texttt{ by } N\})$ we first compute $\theta_m(M\{\mathbb{X} \texttt{ by } N\})$ which gives the preterm $t = a\langle g(x,x), z\rangle\{a\langle x, f(z,y)\rangle \texttt{ by } f(x,x)\}$.

Now, since α-conversion is allowed on preterms, we obtain

$$t =_\alpha a\langle g(x', x'), z\rangle\{a\langle x', f(z,y)\rangle \texttt{ by } f(x,x)\} = t'$$

Now, the computation of $\{\!\{a\langle x', f(z,y)\rangle \texttt{ by } f(x,x)\}\!\}$ gives $\{\rho_1, \rho_2\}$, where $\rho_1 = \{a \triangleright 1, x' \triangleright f(x,x)\}$ and $\rho_2 = \{a \triangleright 2, z \triangleright x, y \triangleright x\}$, and thus, the second step of the application procedure finally gives a set

$$\theta_v(t') = \{g(f(x,x), f(x,x)), x\}$$

As expected, acceptable terms are stable by acceptable substitutions:

Lemma 2. *If t is an acceptable preterm and θ is a substitution acceptable w.r.t $\mathcal{FV}(t)$, then $\theta(t)$ contains only acceptable terms.*

Proof. By induction on t.

- For $t = x$ the result is obvious.
- For $t = f(t_1, \ldots, t_n)$ or $a\langle t_1, \ldots, t_n\rangle$ the property is obvious by i.h.
- If $t = \mu p.t'$, then by α-conversion we can suppose that $Codom(\theta) \cap \mathcal{V}ar(p) = \emptyset$ and thus the result holds by i.h.
- If $t = u\{p \texttt{ by } v\}$, then also by α-conversion we can suppose that $Codom(\theta) \cap \mathcal{V}ar(p) = \emptyset$. By i.h. every $v' \in \theta(v)$ contains only acceptable terms so that by Lemma 1 any $\theta' \in \{\!\{p \texttt{ by } v'\}\!\}$ is an acceptable substitution. Moreover we have that $\theta' \sqcup \theta$ is an acceptable substitution and thus the result holds by i.h. on u.

2 Rewrite rules and reduction relation

This section introduces the precise syntax used to specify rewrite rules in the ERSP formalism as well as the reduction relation associated to them.

Definition 15. *An Expression Reduction System with Patterns (ERSP) is a set of rewrite rules of the form $l \longrightarrow r$ (written also (l, r)) such that:*
- *l and r are well-formed metaterms,*
- *the first symbol (called head symbol) in l is in $\mathcal{F} \cup \mathcal{B}$,*
- *$\mathcal{MV}(r) \subseteq \mathcal{MV}(l)$,*
- *$\mathcal{FV}(r) \subseteq \mathcal{FV}(l)$, and*
- *l contains no occurrence of the pattern-matching constructor.*

Thus for example, the rule $app(\lambda\mathbb{X}.M, N) \longrightarrow M\{\mathbb{X}\ \mathtt{by}\ N\}$ given in the introduction, which generalizes the classical β-rule to the case of patterns, belongs to our framework.

In order to be able to guarantee that no free variable is "generated" during reduction the following notion will be necessary.

Definition 16 (Path condition). *Let M be a term variable and t be a metaterm. We consider all the occurrences $p_1, \ldots, p_n$ of M in t and their corresponding parameter paths $l_1, \ldots, l_n$. A metasubstitution θ is said to have the path condition property for M in t iff:*

$$\forall \widehat{x} \in \mathcal{FV}(\theta_m M), (\forall 1 \leq i \leq n, \widehat{x} \in \theta_m l_i) \vee (\forall 1 \leq i \leq n, \widehat{x} \notin \theta_m l_i)$$

where the notation $\theta_m l$ denotes the set $\bigcup_{\widehat{x} \in l} \theta_m \widehat{x}$.

This notion is extended to rewrite rules by saying that θ has the path condition property for M in (l, r) iff it has the path condition for M in $f(l, r)$, where f is any binary function symbol. This trick is just used to consider a rule as a unique "tree".

The classical example of path condition which is not satisfied for a rewrite rule is given by the η-rule of the λ-calculus (see for example [24, 5]). We give here another version in the same spirit but using patterns:

$$\lambda f(\mathbb{X}).M \longrightarrow M$$

The metasubstitution $\theta = \{\mathbb{X} \triangleright x, M \triangleright x\}$ does not verify the path condition for M in this rule.

We now define the set of "good" substitutions to instantiate rewrite rules. For that we remark that given a rewrite rule $l \longrightarrow r$, the metaterm l does not contain the metasubstitution operator, so that for any metasubstitution θ the term $\theta(l)$ is a singleton.

Definition 17 (Admissible metasubstitution for metaterms/rules). *A metasubstitution θ is admissible for a metaterm t iff*

- *$\theta(t)$ contains only acceptable terms*
- *θ has the path condition for every term metavariable appearing in t.*

A metasubstitution θ is admissible for a rule (l, r) iff θ is admissible for $f(l, r)$, where f is any binary function symbol.

We remark that this definition implies that given a rule (l, r) both $\theta(l)$ and $\theta(r)$ are defined, so in particular all the pattern/term metavariables in l are also in $Dom(\theta_m)$.

Definition 18 (Admissible reduction relation). *Let $\mathcal{R}$ be a ERSP. We say that s rewrites to t, written $s \longrightarrow_{\mathcal{R}} t$ (or $s \xrightarrow{a}_{\mathcal{R}} t$ when the distinction must be done), iff there exists a rewrite rule $(l, r) \in \mathcal{R}$, a well-formed admissible metasubstitution θ for (l, r) and a context C such that $s = C[\theta(l)]$ and $t \in C[\theta(r)]$. This notion can also be defined by induction by the following sentences:*

$$\frac{(l, r) \in \mathcal{R}}{\theta(l) \longrightarrow_{\mathcal{R}} t}\ t \in \theta(r) \qquad\qquad \frac{t \longrightarrow_{\mathcal{R}} u}{C[t] \longrightarrow_{\mathcal{R}} C[u]}\ C \text{ is a context}$$

Even if the relation $\longrightarrow_{\mathcal{R}}$ is defined on any kind of terms, the reduction can only take place on acceptable subterms.

As expected, the relation reduction enjoys preservation of free variables.

Lemma 3. *If $l \longrightarrow r$ is a rewrite rule and θ is an admissible metasubstitution for $l \longrightarrow r$, then $\forall s \in \theta(r) \forall a \forall i$ we have $\mathcal{FV}^i_a(s) \subseteq \mathcal{FV}^i_a(\theta(l))$.*

Proof. Let us take any $s \in \theta(r)$ and let us suppose that $x \in \mathcal{FV}^i_a(s)$. Two cases are possible:

1. If $x \in \mathcal{FV}(r)$, then r is not well-formed leading to a contradiction.
2. If $x \notin \mathcal{FV}(r)$, then x was introduced by θ and thus in particular $x \in \mathcal{FV}^i_a(\theta M)$ for some term metavariable M appearing in r. Moreover, there exists a position p_r of r s.t $r|_{p_r} = M$, l_r is the parameter path of r at p_r and $x \notin \theta(l_r)$. By definition of rewriting rule, M appears also in l so that there is a position p_l in l such that $l|_{p_l} = M$. Let us take the parameter path l_l of l at p_l. We have the following cases:
   - $x \in \theta(l_l)$ is not possible since θ has the path condition for M in (l, r).
   - $x \notin \theta(l_l)$:
     (a) If $a \in \theta(l_l)$, then $\mathcal{FV}^i_a(\theta l) = \mathcal{FV}(\theta l)$ by remark 1 and thus the result obviously holds since $x \in \mathcal{FV}(\theta(M)) \subseteq \mathcal{FV}(\theta l)$.
     (b) If $a \notin \theta(l_l)$, then since $a \notin \mathcal{FV}(l) \cup \mathcal{FV}(r)$ by definition of well-formed metaterm, we finally have $x \in \mathcal{FV}^i_a(\theta(M)) \subseteq \mathcal{FV}^i_a(\theta l)$.

Lemma 4 (Preservation of localized free variables).
If $s \longrightarrow_{\mathcal{R}} t$, then $\forall a, \forall i, \mathcal{FV}^i_a(t) \subseteq \mathcal{FV}^i_a(s)$.

Proof. By induction on the definition of $s \longrightarrow_{\mathcal{R}} t$ using Lemma 3.

Corollary 1 (Preservation of free variables). *If $s \longrightarrow_{\mathcal{R}} t$, then $\mathcal{FV}(t) \subseteq \mathcal{FV}(s)$.*

Proof. Using Lemma 4 with a choice variable $a \notin \mathcal{FV}(s) \cup \mathcal{FV}(t)$ and Remark 1.

We can finally conclude this section by an essential property of reduction.

Lemma 5 (Preservation of acceptability). *If s is an acceptable term and $s \longrightarrow_{\mathcal{R}} t$, then t is also acceptable.*

Proof. By induction on the definition of $s \longrightarrow_{\mathcal{R}} t$ using Lemma 4.

3 A complete example

We consider in this section the well known higher-order function *map* which takes a function f and a list l and returns the result of applying f to each element of l. This function can be specified as the following Ocaml [31] program:

```
#let rec map(f,l) =
  match l with
    | Nil -> Nil
    | Cons(h,t) -> Cons(f(h),map(f,t));;
```

It can also be specified as the following ERSP rewrite rule:

$$\begin{array}{l} map(\mu\mathbb{X}.F, L) \longrightarrow \\ \quad a\langle nil, cons(F\{\mathbb{X}\ \texttt{by}\ h\}, map(\mu\mathbb{X}.F, t))\rangle\{a\langle nil, cons(h,t)\rangle\ \texttt{by}\ L\} \end{array}$$

Let us see how this previous implementation of *map* works on a concrete example. Suppose that we represent natural numbers with constructors 0 and s and let us consider $pred =_{def} \mu b\langle 0, s(n)\rangle.b\langle 0, n\rangle$ in order to denote the predecessor function on natural numbers. Using the metasubstitution $\theta = (\theta_m, \theta_v)$, where $\theta_m = \{\mathbb{X} \triangleright b\langle 0, s(n)\rangle, F \triangleright b\langle 0, n\rangle, L \triangleright cons(0, cons(s(s(0)), nil))\}$ and $\theta_v = \emptyset$ to fire the previous rewrite rule, we can construct the following derivation:

$$\begin{array}{ll} t_1 = map(pred, cons(0, cons(s(s(0)), nil))) & \longrightarrow \\ t_2 = cons(0, map(pred, cons(s(s(0)), nil))) & \longrightarrow \\ t_3 = cons(0, cons(s(0), map(pred, nil))) & \longrightarrow \\ t_4 = cons(0, cons(s(0), nil)) & \end{array}$$

Indeed, the term t_1 is an instance of the left-hand side of the previous rule. In order to obtain t_2 we have to apply θ to the right-hand side of the previous rule. For that, we first instantiate $\{a\langle nil, cons(h,t)\rangle\ \texttt{by}\ L\}$ with θ_m, then since $\theta_v(\theta_m(L)) = \theta_m(L)$, we can compute the pattern-matching operation $\{\!\!\{a\langle nil, cons(h,t)\rangle\ \texttt{by}\ cons(0, cons(s(s(0)), nil))\}\!\!\}$. We then obtain a substitution $\theta'_v = \{a \triangleright 2, h \triangleright 0, t \triangleright cons(s(s(0)), nil)\}$. Now, we have to instantiate $a\langle nil, cons(F\{\mathbb{X}\ \texttt{by}\ h\}, map(\mu\mathbb{X}.F, t))\rangle$ with θ_m, then proceed with the application of θ'_v to this last instantiation. The only delicate part is the one concerning the submetaterm $F\{\mathbb{X}\ \texttt{by}\ h\}$. We have $\theta_m(F\{\mathbb{X}\ \texttt{by}\ h\}) = b\langle 0, n\rangle\{b\langle 0, s(n)\rangle\ \texttt{by}\ h\} = t'$ and $\theta'_v(t') = b\langle 0, n\rangle\{\!\!\{b\langle 0, s(n)\rangle\ \texttt{by}\ 0\}\!\!\} = 0$.

The reader may verify that this sequence of operations finally leads to the term t_2. Similarly, we then obtain t_3 and t_4.

4 Towards a subclass of confluent ERSP

The following two sections are devoted to the study of confluence for a certain class of ERSP which are called the *orthogonal l-constructor* ERSP, and a certain class of terms, which are called *l-constructor deterministic terms*. Intuitively, an orthogonal ERSP is *left-linear* and *not overlapping*. The need of orthogonality for confluence in first and higher-order rewrite systems is well-known [2]. A constructor ERSP is a system $\mathcal{R}$ where the set of function symbols is partitioned into two different subsets, namely, the set of *constructors*, which cannot be reduced, and the set of *defined symbols*, which cannot be matched. As an example, let us consider the following system which is not a constructor ERSP.

$$\mathcal{R} = \begin{cases} a \longrightarrow\ b \\ app(\mu a.c, M) \longrightarrow\ c\{a \texttt{ by } M\} \end{cases}$$

The term $app(\mu a.c, a)$ can be reduced to both $app(\mu a.c, b)$ and c which are not joinable. Thus, $\mathcal{R}$ turns out to be non confluent.

Unfortunately, *orthogonal l-constructor* ERSP do not immediately guarantee confluence as the rule $app(\lambda\mathbb{X}.M, N) \longrightarrow\ M\{\mathbb{X} \texttt{ by } N\}$ shows: the term $t = app(\lambda a\langle x, y\rangle.a\langle 0, 1\rangle, 3)$ has two non-joinable reducts 0 and 1 by this unique rule.

The reason is that t contains two "overlapping" patterns x and y inside the choice pattern $a\langle x, y\rangle$. The failure of the confluence property in this case is completely natural since the term t corresponds, informally, to a "non-orthogonal" first-order rewriting system. It is then clear that we have to get rid of this class of terms in order to get a confluence result, this will be done by introducing the notion of *l-constructor deterministic* terms.

We are know ready to give a formal definition of all these notions.

Definition 19 (L-constructor system). *$\mathcal{R}$ is said to be l-constructor iff*

- *The set $\mathcal{F}$ of function symbols can be partitioned into two sets $\mathcal{F}_c$ and $\mathcal{F}_d$, called respectively constructors and defined symbols, such that:*
  - *Each defined symbol is the head of some left-hand side of $\mathcal{R}$.*
  - *All the function symbols in metapatterns of $\mathcal{R}$ are constructors.*
- *For every rule $(l, r) \in \mathcal{R}$, both l and r are p-linear metaterms.*

In the rest of this section we will use the notation C or C_i to denote the elements of $\mathcal{F}_c$.

The system $\mathcal{R}_1 = \{\beta_{PM}\} \cup \{0 + N \longrightarrow\ N, s(M) + N \longrightarrow\ s(M + N)\}$ is l-constructor. The system $\mathcal{R}_2 = \{\mu f(\mathbb{X}).M \longrightarrow\ M, f(0) \longrightarrow\ 0\}$ is not l-constructor since the function symbol f appears as the head symbol of some rule and inside a metapattern of $\mathcal{R}$. The system $\mathcal{R}_3 = \{\mu f(\mathbb{X}, \mathbb{X}).0 \longrightarrow\ 0\}$ is not l-constructor since $\mu f(\mathbb{X}, \mathbb{X}).0$ is not p-linear.

Definition 20 (L-constructor (meta)patterns and (meta)terms). *Given a l-constructor system $\mathcal{R}$, we say that a metapattern is l-constructor iff it is linear and all its function symbols are constructors of $\mathcal{R}$. A l-constructor metaterm contains only l-constructor metapatterns.*

As an example concerning our previous system $\mathcal{R}_1$, we can observe that the metapattern $s(\mathbb{X})$ is l-constructor but $plus(\mathbb{X}, \mathbb{Y})$ is not since the symbol $plus$ is not a constructor function symbol.

Even if Definition 19 depends on a given l-constructor system $\mathcal{R}$ we will make an abuse of notation by just writing l-constructor metapattern/metaterm instead of $\mathcal{R}$ l-constructor metapattern/metaterm.

Definition 21 (L-constructor metasubstitution). *Given a l-constructor system $\mathcal{R}$, a metasubstitution θ is said to be a l-constructor w.r.t. a set S of (meta)variables iff for any $\widehat{x} \in S$, $\theta(\widehat{x})$ is l-constructor and θ is said to be l-constructor iff θ is l-constructor w.r.t $Dom(\theta)$.*

We remark that if p is a l-constructor pattern and $\{\!\{p \texttt{ by } t\}\!\}$ is defined then all its elements are l-constructor substitutions.

Definition 22 (L-constructor reduction relation). *If $\mathcal{R}$ is a l-constructor system, we say that s constructor rewrites to t (written $s \xrightarrow{c}_{\mathcal{R}} t$) iff there exists a rewrite rule $(l, r) \in \mathcal{R}$, a well-formed l-constructor metasubstitution θ admissible for (l, r) and a context C such that $s = C[\theta(l)]$ and $t \in C[\theta(r)]$.*

As an example, given the previous system $\mathcal{R}_1$, we have $plus(0,0) \xrightarrow{c}_{\mathcal{R}_1} 0$ but we do not have $t = app(\lambda plus(0,0).3, plus(0,0)) \xrightarrow{c}_{\mathcal{R}_1} 3$ (even if we have $t \xrightarrow{a}_{\mathcal{R}_1} 3$) since the term t is not a l-constructor term.

One can remark that whenever t is a l-constructor preterm and θ is a l-constructor substitution w.r.t. $\mathcal{FV}(t)$, then $\theta(t)$ contains only l-constructor terms. Also as expected, l-constructor terms are preserved during reduction.

Lemma 6 (Preservation of l-constructor terms). *If $\mathcal{R}$ is l-constructor, s is l-constructor and $s \xrightarrow{c}_{\mathcal{R}} t$, then t is l-constructor.*

Proof. We prove this statement by induction on $s \xrightarrow{c}_{\mathcal{R}} t$.

- If $s = \theta(l)$ and $t \in \theta(t)$ for a rule (l, r), the result holds by hypothesis.
- If $s = \mu p.s'$ and $t = \mu p.t'$ with $s' \xrightarrow{c}_{\mathcal{R}} t'$, let us consider a pattern p' appearing in t. Only two cases are possible:
  1. p' is a subpattern of p and then the result holds since s is supposed to be a l-constructor term.
  2. p' appears in t' and then the result holds by i.h.
- The other cases are obvious by i.h.

Definition 23 (Left linear systems). *A rewrite rule $l \longrightarrow r$ is said to be left linear iff l contains at most one occurrence of any term metavariable. A system is left linear if all its rule are left linear.*

As an example, the rule $f(M, M) \longrightarrow 3$ is not left linear while $f(M) \longrightarrow g(M, M)$ and $\mu x.f(x, x) \longrightarrow 0$ are.

Definition 24 (Redexes and overlapping redexes). *A term t is said to be a redex if it is an instance of some left-hand side of rule that is $t = \theta(l)$ for some rule (l, r). A rewrite system is said to be non-overlapping iff*

- *Whenever a redex $\theta(l_j)$ contains (not necessarily properly) another redex $\theta'(l_i)$, then $\theta'(l_i)$ must be contained in $\theta(M)$ for some term metavariable M of l_j.*
- *Likewise whenever a redex $\theta(l)$ properly contains another redex instance $\theta'(l)$ of the same rule.*

Definition 25 (Orthogonal systems). *A rewrite system $\mathcal{R}$ is said to be orthogonal iff $\mathcal{R}$ is left-linear and non-overlapping.*

As an example, the system $\{f(\mu x.x) \longrightarrow 0, \mu\mathbb{X}.y \longrightarrow 1\}$ is overlapping. The system $\{f(\mu\mathbb{X}.M) \longrightarrow 0, \lambda\mathbb{Z}.N \longrightarrow g(2)\}$ is orthogonal.

We now introduce *l-constructor deterministic terms* for which the class of orthogonal l-constructor ERSP will be confluent. Let us start by the following notion.

Definition 26 (Overlapping patterns). *Two patterns p and q are said to be overlapping iff $\exists$ a term t s.t. both $\{\!\{p \texttt{ by } t\}\!\}$ and $\{\!\{q \texttt{ by } t\}\!\}$ are defined.*

The patterns $f(\_, x)$ and $f(y, g(0))$ are overlapping. Also $a\langle 0, s(x)\rangle$ and $b\langle s(0), s(s(\_))\rangle$ are overlapping.

Definition 27 (Deterministic patterns/terms). *The set of deterministic patterns is defined to be the smallest subset of linear patterns containing wildcard and variables, closed by algebraic and contraction patterns, and such that if $p_1, \ldots, p_n$ are deterministic and for all $i \neq j$ the patterns p_i and p_j are not overlapping, then $a\langle p_1, \ldots, p_n \rangle$ is deterministic. A term t is said to be a deterministic term iff t is acceptable and for every pattern p appearing in t, p is deterministic.*

Thus for example, $b\langle s(0), s(s(\_))\rangle$ is deterministic but $b\langle s(0), s(\_)\rangle$ is not.

Remark 2. We remark that this definition implies that whenever p is a deterministic pattern, then there exists at most one substitution θ belonging to $\{\!\{p \texttt{ by } t\}\!\}$.

Definition 28 (Deterministic metasubstitution for metaterms/rules). *A metasubstitution θ is said to be deterministic for a metaterm t iff*
- *θ is admissible for t,*
- *$\theta(t)$ contains only one deterministic term,*

Finally, θ is deterministic for a rule (l, r) iff θ is a deterministic for $f(l, r)$, where f is any binary function symbol.

A *metasubstitution* θ is *deterministic* iff $\theta\widehat{x}$ is deterministic for every $\widehat{x} \in Dom(\theta)$. Deterministic terms are stable by deterministic substitutions.

Lemma 7. *If p is a deterministic pattern, t is a deterministic term and $\theta = \{\!\{p \texttt{ by } t\}\!\}$ is defined, then θ is a deterministic substitution.*

Proof. We prove this statement by induction on p.

- If $p = \_$ or $p = x$, the result is obvious.
- If $p = f(p_1, \ldots, p_n)$, by definition of $\{\!\{ \texttt{ by } \}\!\}$ we must have $t = f(t_1, \ldots, t_n)$. and all the $\theta_i = \{\!\{p_i \texttt{ by } t_i\}\!\}$ are defined. By i.h., since all the p_i and all the t_i are deterministic, all the θ_i are deterministic. Hence the result holds.
- If $p = @(p_1, \ldots, p_n)$, by definition of $\{\!\{ \texttt{ by } \}\!\}$ we know that for all i $\theta_i = \{\!\{p_i \texttt{ by } t\}\!\}$ is defined. Moreover since p_i and t are deterministic, by i.h., we know that θ_i is deterministic. Hence the result trivially holds.
- If $p = a\langle p_1, \ldots, p_n \rangle$, then by definition of $\{\!\{ \texttt{ by } \}\!\}$ and the hypothesis there exists a unique i such that $\theta_i = \{\!\{p_i \texttt{ by } t\}\!\}$ is defined. By i.h., we know that θ_i is deterministic. Hence the result holds.

Definition 29 (Deterministic reduction relation). *Given a system $\mathcal{R}$, we say that s deterministicly rewrites to t (written $s \xrightarrow{d}_{\mathcal{R}} t$) iff there exists a rewrite rule $(l, r) \in \mathcal{R}$, a well-formed deterministic metasubstitution θ for (l, r) and a context C such that $s = C[\theta(l)]$ and $s = C[\theta(r)]$.*

From now on we use the notation $\xrightarrow{c,d}_{\mathcal{R}}$ to denote $\xrightarrow{c}_{\mathcal{R}} \cap \xrightarrow{d}_{\mathcal{R}}$. As expected, orthogonal systems allows us to preserve deterministic terms.

Lemma 8 (Preservation of deterministic terms). *Given an orthogonal system $\mathcal{R}$, if s is deterministic and $s \xrightarrow{d}_{\mathcal{R}} t$, then t is deterministic.*

Proof. Since s is in particular acceptable, then we know, by Lemma 5, that t is also acceptable. We prove that each patterns p appearing in t is deterministic by induction on the definition of $s \xrightarrow{d}_{\mathcal{R}} t$.

– If $s = \theta(l)$ and $t = \theta(r)$ for some $l \longrightarrow\ r \in \mathcal{R}$, then the result holds by hypothesis.
– If $s = \mu p.s'$ and $t = \mu p.t'$ with $s' \xrightarrow{d}_{\mathcal{R}} t'$, let us consider a pattern p' appearing in t. Only two cases are possible:
  1. p' is a subpattern of p and then the result holds since s is supposed to be deterministic.
  2. p' appears in t' and then the result holds by i.h.
– The other cases are obvious by i.h.

5 The confluence proof

This section is dedicated to show that the relation $\xrightarrow{c,d}_{\mathcal{R}}$ is confluent for orthogonal l-constructor ERSP, that is, orthogonal l-constructor ERSP are confluent on l-constructor deterministic terms. This confluence property can only be proved on the set of acceptable l-constructor deterministic terms. The proof uses a technique due to Tait and Martin-Löf [3] and can be summarized in four steps:

- We define a parallel reduction relation denoted $\overset{c,d}{\ggg}$.
- We prove that $\overset{c,d}{\ggg}^*$ and $\xrightarrow{c,d}^*$ are the same relation.
- We show, using Takahashi terms [33], that $\overset{c,d}{\ggg}$ has the diamond property on the set of acceptable l-constructor deterministic terms.
- We conclude by the fact that the diamond property implies confluence.

In order to define the $\overset{c,d}{\ggg}$ reduction relation, we first need to extend relations on terms to relations on substitutions.

Notation 1 Given any relation $\rightsquigarrow$ between terms and given two metasubstitutions θ and θ', we write $\theta \rightsquigarrow \theta'$ when $Dom(\theta) = Dom(\theta')$, θ and θ' coincide on the sets of choice and pattern variables, and for every $M \in Dom(\theta)$ we have $\theta(M) \rightsquigarrow \theta'(M)$ and $x \in Dom(\theta)$ we have $\theta(x) \rightsquigarrow \theta'(x)$.

In order to relate two sets S and T of metaterms/metasubstitutions via any binary relation $\rightsquigarrow$, we will write $S \rightsquigarrow T$ iff $\forall s \in S, \exists t \in T, s \rightsquigarrow t$ and $\forall t \in T, \exists s \in S, s \rightsquigarrow t$. For any unary relation Φ on metaterms/metasubstitutions, we will write $\Phi(S) = \bigcup_{s \in S} \Phi(s)$.

We can now define the simultaneous relation as follows:

Definition 30 (The simultaneous relation $\overset{a}{\ggg}$).

- *$x \overset{a}{\ggg} x$.*
- *If $s_1 \overset{a}{\ggg} s'_1, \ldots, s_m \overset{a}{\ggg} s'_m$, then $f(s_1, \ldots, s_m) \overset{a}{\ggg} f(s'_1, \ldots, s'_m)$.*
- *If $s \overset{a}{\ggg} s'$, then $\mu p.s \overset{a}{\ggg} \mu p.s'$.*
- *If $s_1 \overset{a}{\ggg} s'_1, \ldots, s_m \overset{a}{\ggg} s'_m$, then $a\langle s_1, \ldots, s_m\rangle \overset{a}{\ggg} a\langle s'_1, \ldots, s'_m\rangle$.*
- *If $\theta \overset{a}{\ggg} \rho$, then $\theta(l) \overset{a}{\ggg} \rho(r)$ for any rule $l \longrightarrow\ r$ such that θ is admissible for it.*

We will denote $s \overset{c}{\ggg} t$ (resp. $s \overset{d}{\ggg} t$) if the metasubstitution θ used in the Definition 30 is not only admissible but also l-constructor (resp. deterministic).

Remark 3. Given $\ggg \in \{\overset{a}{\ggg}, \overset{c}{\ggg}, \overset{d}{\ggg}\}$, then for every term s we have $s \ggg s$. Also, if $s \ggg s'$ and s is not a redex, then the root symbols of s and s' are the same. Moreover, if also $s = \mathcal{G}(s_1, \ldots, s_m)$, then $s' = \mathcal{G}(s'_1, \ldots, s'_m)$, where $s_i \ggg s'_i$, for $\mathcal{G}$ a function symbol, a case or a binder symbol.

We are now ready to proceed with the second step of Tait and Martin-Löf's technique which consists in showing that the reflexive-transitive closures of $\xrightarrow{c,d}$ and $\overset{c,d}{\ggg}$ are the same relation.

Lemma 9. *Let $\mathcal{R}$ be a l-constructor ERSP, p a l-constructor pattern and t a term such that $t \xrightarrow{c}{}^{*}_{\mathcal{R}} t'$. If $\{\!\{p \texttt{ by } t\}\!\}$ is defined, then $\{\!\{p \texttt{ by } t'\}\!\}$ is also defined and $\{\!\{p \texttt{ by } t\}\!\} \xrightarrow{c}{}^{*}_{\mathcal{R}} \{\!\{p \texttt{ by } t'\}\!\}$.*

Proof. We have to show that $\{\!\{p \texttt{ by } t'\}\!\}$, that $\forall \theta \in \{\!\{p \texttt{ by } t\}\!\} \exists \theta' \in \{\!\{p \texttt{ by } t'\}\!\} \theta \xrightarrow{c}{}^{*}_{\mathcal{R}} \theta'$ and $\forall \theta' \in \{\!\{p \texttt{ by } t'\}\!\} \exists \theta \in \{\!\{p \texttt{ by } t\}\!\} \theta \xrightarrow{c}{}^{*}_{\mathcal{R}} \theta'$. The proof proceeds by induction on p.

- If $p = x$ or $p = \_$ the result is obvious.
- If $p = C(p_1, \ldots, p_n)$ then by definition of $\{\!\{ \texttt{ by } \}\!\}$ we must have $t = C(t_1, \ldots, t_n)$ with $\{\!\{p_i \texttt{ by } t_i\}\!\}$ defined for all i. Since we know that no rule in $\mathcal{R}$ is headed by a constructor symbol, then the reduction of t cannot take place at the root of t. Thus, $t' = C(t'_1, \ldots, t'_n)$ with $t_i \xrightarrow{c}{}^{*}_{\mathcal{R}} t'_i$ for every i. By i.h. we know that all the $\{\!\{p_i \texttt{ by } t'_i\}\!\}$ are defined and that $\{\!\{p_i \texttt{ by } t_i\}\!\} \xrightarrow{c}{}^{*} \{\!\{p_i \texttt{ by } t'_i\}\!\}$ for every i. Let us first show that $\{\!\{p \texttt{ by } t'\}\!\}$ is defined. Take one θ'_i in each $\{\!\{p \texttt{ by } t_i\}\!\}$. We have just to show that $\bigsqcup_i \theta'_i$ is defined but this immediately holds since p is a linear pattern. The fact that $\{\!\{p \texttt{ by } t\}\!\} \xrightarrow{c}{}^{*}_{\mathcal{R}} \{\!\{p \texttt{ by } t'\}\!\}$ is obvious by i.h.
- If $p = @(p_1, \ldots, p_n)$ or $p = a\langle p_1, \ldots, p_n\rangle$ then we proceed as in the previous case.

The previous Lemma does not holds if the pattern p is not *l-constructor*. Indeed, let $\mathcal{R} = \{a \longrightarrow b\}$ and let us take the non-linear pattern $p = f(x,x)$. We obviously have that $\{\!\{f(x,x) \texttt{ by } f(a,a)\}\!\}$ is defined and $f(a,a) \xrightarrow{c}_{\mathcal{R}} f(a,b)$ but $\{\!\{f(x,x) \texttt{ by } f(a,b)\}\!\}$ is not defined. Besides that, let us take the non-constructor pattern $p = a$. The substitution $\{\!\{a \texttt{ by } a\}\!\}$ is defined and $a \xrightarrow{c}_{\mathcal{R}} b$ but $\{\!\{a \texttt{ by } b\}\!\}$ is not defined.

Corollary 2. *Let $\mathcal{R}$ be a l-constructor ERSP. Let p be a l-constructor metapattern and t be a metaterm. Let ρ and ρ' be two metasubstitutions such that $\rho(p)$ is a l-constructor pattern and $\rho(t) \xrightarrow{c}{}^{*}_{R} \rho'(t)$. If $\{\!\{\rho(p) \texttt{ by } \rho(t)\}\!\}$ is defined, then $\{\!\{\rho'(p) \texttt{ by } \rho'(t)\}\!\}$ is also defined and $\{\!\{\rho(p) \texttt{ by } \rho(t)\}\!\} \xrightarrow{c}{}^{*}_{\mathcal{R}} \{\!\{\rho'(p) \texttt{ by } \rho'(t)\}\!\}$.*

Proof. Obvious using Lemma 9 since $\theta(p) = \rho(p)$ by definition.

We are now ready to show that the reduction of metasubstitutions is stable by application:

Lemma 10. *Let $\theta = (\theta_m, \theta_v)$ be a metasubstitution, t a well-formed metaterm and S a set of choice/usual variables. Let also suppose that θ verifies the path condition for every term metavariable M in t. Then, there exist $\theta' = (\theta'_m, \theta_v)$ and a well-formed metaterm t' such that $\theta_m(t) =_\alpha \theta'_m(t')$, where t' only differs from t on the set of (bound) usual/choice variables, and none bound variable of $\theta'_m(t')$ belongs to S.*

Proof. The path condition guarantees that for every term metavariable M in t there is a *unique* set of choice/usual variables $\mathcal{V}_M$ such that $\widehat{x} \in \mathcal{V}_M$ iff $\widehat{x} \in \theta_m(\mathcal{PP}(M,p)$ for *any* position p such that $t|_p = M$. Also, we have for every pattern metavariable $\mathbb{X}$ a *unique* set of choice/usual variables $\mathcal{V}_{\mathbb{X}} = \mathcal{V}ar(\theta_m(\mathbb{X}))$. Let also consider the set of bound choice/usual variables $\mathcal{A}$ of t. We define a renaming ρ from $\bigcup_{M \in t} \mathcal{V}_M \cup \bigcup_{\mathbb{X} \in t} \mathcal{V}_{\mathbb{X}} \cup \mathcal{A}$ to a new set $\mathcal{V}'$ such that $\mathcal{V}' \cap S = \emptyset$. Now, we construct t' by (first-order) mapping every (meta)variable $\widehat{x}$ of $\mathcal{A}$ to its correspondent image $\rho(\widehat{x})$. We then construct θ'_m as follows: every pattern metavariable $\mathbb{X}$ is mapped to $\rho(\theta_m(\mathbb{X}))$ and every term metavariable M is mapped to $\rho(\theta_m(M))$.

Lemma 11. *Let $\theta = (\theta_m, \theta_v)$ be a well-formed metasubstitution and t a well-formed metaterm. Let also suppose that θ verifies the path condition for every term metavariable M in t. Then, there exist $\theta' = (\theta'_m, \theta'_v)$ and a well-formed metaterm t' such that $\theta(t) =_\alpha \theta'(t')$, where t' is as in the previous lemma, and the bound variables of $\theta'_m(t')$ do not belong to $Dom(\theta'_v) \cup Codom(\theta'_v)$.*

Proof. We first use Lemma 10 to construct θ'_m and t' w.r.t the set $Dom(\theta_v) \cup Codom(\theta_v)$. Then we take $\theta'_v = \theta_v$. Hence the result follows.

Lemma 12. *Let $\mathcal{R}$ be an l-constructor ERSP and θ, ρ two well-formed l-constructor metasubstitutions such that $\theta \xrightarrow{c}{}^{*}_{\mathcal{R}} \rho$. If for any l-constructor metaterm t such that θ has the path condition for t, $\theta(t)$ is defined and contains only acceptable l-constructor terms, then the same happens for $\rho(t)$ and $\theta(t) \xrightarrow{c}{}^{*}_{\mathcal{R}} \rho(t)$.*

Proof. By Lemma 11 we can suppose that θ, ρ and t satisfy the statement concerning the bound variables. We reason by induction the metaterm t.

- For $t = x$ or $t = M$ the result is straightforward by hypothesis. In particular, we use the fact that θ is well-formed in the second case.
- For $t = f(t_1, \ldots, f_n)$ or $t = a\langle t_1, \ldots, t_n \rangle$, the result is obvious by the i.h.
- For $t = \mu p.t'$, the remark done at the beginning of the proof allows us to use the i.h. directly on t'. Indeed, no capture of variables is possible when applying θ_v to $\mu \theta_m(p).\theta_m(t')$.
- $t = u\{p \texttt{ by } v\}$ we reason as in the previous case.

Lemma 12 allows us to obtain the following fundamental property.

Lemma 13. *Let $\mathcal{R}$ be a l-constructor ERSP. If $\theta \xrightarrow{c}{}^{*}_{\mathcal{R}} \rho$, and θ is a well-formed l-constructor admissible metasubstitution for (l, r), then $\theta(l) \xrightarrow{c}{}^{*}_{\mathcal{R}} \rho(r)$ and ρ is admissible for (l, r).*

Proof. We first remark that θ and ρ are identical on the set of pattern metavariables. In order to show that ρ is admissible for (l, r) we have to prove that:

1. $\rho(l)$ and $\rho(r)$ both contain only acceptable terms. By hypothesis θ is admissible for (l, r) and thus both $\theta(l)$ and $\theta(r)$ contain only acceptable l-constructor terms. Hence the statement holds by Lemma 12.
2. The metasubstitution ρ satisfies the path condition for any term variable M in (l, r). Let us take a term variable M. We know that θ satisfies the path condition for M in (l, r), that is, if the p_i's are the occurrences of M in (l, r) and the l_i' are their associated parameter paths:

$$\forall \widehat{x} \in \mathcal{FV}(\theta(M)), (\forall 1 \leq i \leq n, \widehat{x} \in \theta\ l_i) \vee (\forall 1 \leq i \leq n, \widehat{x} \notin \theta\ l_i)$$

   Let us take $\widehat{x} \in \mathcal{FV}(\rho(M))$. Then by Corollary 1 $\widehat{x} \in \mathcal{FV}(\theta(M))$ and hence the result holds.
3. For any $\widehat{x} \in Dom(\rho)$, $\rho(\widehat{x})$ is a l-constructor term. Take any $\widehat{x} \in Dom(\rho)$. By hypothesis $Dom(\theta) = Dom(\rho)$ and $\theta(\widehat{x})$ is l-constructor, so that the result holds by Lemma 6 since $\theta(\widehat{x}) \xrightarrow{c}{}^{*}_{\mathcal{R}} \rho(\widehat{x})$.

We proceed now by induction on $\mu = \Sigma_{i=1 \ldots n} |\theta(x_i) \xrightarrow{c}{}^{*}_{\mathcal{R}} \rho(x_i)|$, where $Dom(\theta_v) = \{x_1, \ldots, x_n\}$.

- If $\mu = 0$, then $\rho = \theta$ and the property is trivial.

- If $\mu > 0$, then there is some i such that $\theta(x_i) \xrightarrow{c}^{*}_{\mathcal{R}} s \xrightarrow{c}_{\mathcal{R}} \rho(x_i)$. We define ξ as (ξ_m, ξ_v), where $\xi_m = \theta_m$, $\xi_v(x_j) = \rho_v(x_j)$ if $j \neq i$ and $\xi_v(x_i) = s$. We then have $\theta(l) \xrightarrow{c}^{*}_{\mathcal{R}} \xi(r)$ by i.h. Since $\xi \xrightarrow{c}^{*}_{\mathcal{R}} \rho$ holds by definition, then we can conclude $\xi(r) \xrightarrow{c}^{*}_{\mathcal{R}} \rho(r)$ by Lemma 12. Thus we obtain $\theta(l) \xrightarrow{c}^{*}_{\mathcal{R}} \rho(r)$. We obtain that ρ is admissible by the previous point.

We are now able to conclude the second step of our confluence proof by claiming:

Theorem 1 ($\longrightarrow_{\mathcal{R}}$ implies $\ggg$). *Let $\mathcal{R}$ be a ERSP and let s be a term and $\rightsquigarrow$ be a relation in $\{\xrightarrow{a}_{\mathcal{R}}, \xrightarrow{c}_{\mathcal{R}}, \xrightarrow{d}_{\mathcal{R}}\}$. If $s \rightsquigarrow t$ then $s \ggg t$, where $\ggg$ is the corresponding relation corresponding to $\rightsquigarrow$.*

Proof. Let $s \rightsquigarrow s'$. We prove $s \ggg s'$ by induction on the definition of $\rightsquigarrow$.

- $s = \theta(l)$ and $s' \in \theta(r)$. Since $\ggg$ is reflexive, then we have $\theta \ggg \theta$ and thus $s = \theta(l) \ggg s' \in \theta(r)$ by Definition 30.
- For all the other cases the statement holds by i.h.

Theorem 2 ($\overset{c}{\ggg}$ implies $\xrightarrow{c}^{*}_{\mathcal{R}}$). *Let $\mathcal{R}$ be a l-constructor ERSP and let s be a term. If $s \overset{c}{\ggg} t$ then $s \xrightarrow{c}^{*}_{\mathcal{R}} t$.*

Proof. By induction on the structure of s.

- If s is a variable then $s \xrightarrow{c}^{*}_{\mathcal{R}} s'$ immediately holds.
- Otherwise, we have two different cases:
  - If s is an instance $\theta(l)$ of the left-hand side l of some rule $l \longrightarrow r$ and $s' \in \rho(r)$, where $\theta \overset{c}{\ggg} \rho$, then, since $\theta_v(y)$ is smaller than s for every free usual variable y of l, we have by i.h. that $\theta_v(y) \xrightarrow{c}^{*}_{\mathcal{R}} \rho_v(y)$. We can conclude $s = \theta(l) \xrightarrow{c}^{*}_{\mathcal{R}} s' \in \rho(r)$ by Lemma 13 since by hypothesis θ is admissible for (l, r) and θ is obviously well-formed.
  - In all the other cases the statement holds by i.h.

We remark that the previous lemma does not holds in general as suggested by the example immediately after Lemma 9.

Corollary 3. *Let $\mathcal{R}$ be an l-constructor ERSP and let s be a term. Then $s \overset{c}{\ggg}^{*} s'$ iff $s \xrightarrow{c}^{*}_{\mathcal{R}} s'$.*

We are now going to prove the diamond property for the relation $\overset{c,d}{\ggg}$. For that, we associate a term $\#(s)$ to every term s such that every time $s \overset{c,d}{\ggg} s'$, for some s', we automatically deduce $s' \overset{c,d}{\ggg} \#(s)$. Thus, given two different terms s' and s'' such that $s \overset{c,d}{\ggg} s'$ and $s \overset{c,d}{\ggg} s''$, we obviously obtain a unique term $\#(s)$ which allows us to close the diagram with $s' \overset{c,d}{\ggg} \#(s)$ and $s'' \overset{c,d}{\ggg} \#(s)$. The diamond property will immediately follow.

Definition 31 (Takahashi terms $\#(s)$). *Given a ERSP $\mathcal{R}$ and a term s we define its associated takahashi term $\#(s)$ by induction as follows:*
- *If $s = x$, then $\#(x) = x$.*
- *If $s = f(s_1, \ldots, s_m)$ (resp. $s = \mu p.s'$ or $a\langle s_1, \ldots, s_m \rangle$) and s is not a deterministic instance of a left-hand side of rule, then $\#(s) = f(\#(s_1), \ldots, \#(s_m))$ (resp. $\#(s) = \mu p.\#(s')$ and $\#(s) = a\langle \#(s_1), \ldots, \#(s_m) \rangle$).*

- *If s is an instance $\theta(l)$ of the left-hand side l of some rule $l \longrightarrow r$, where $\theta = (\theta_m, \theta_v)$ is deterministic for (l, r), then $\#(s) = \#(\theta)(r)$, where $\#(\theta) = (\#(\theta)_m, \#(\theta)_v)$ verifies $\#(\theta)_m(M) = \#(\theta_m(M))$ and $\#(\theta)_v(y) = \#(\theta_v(y))$.*

We can show by induction on terms the following two properties about $\ggg$.

Lemma 14. *The term $\#(s)$ is uniquely defined in every orthogonal ERSP.*

Proof. By induction on s. All the cases can be shown by i.h. whenever s is not a deterministic instance of the left-hand side of a rule (l, r).

If s is a deterministic instance of a left-hand side of rule, we first remark that this rule must be unique since $\mathcal{R}$ is orthogonal. So let us call (l, r) this rule and θ this instance. We have $s = \theta(l)$ and for any pattern p appearing in s, p is a deterministic pattern. We also reason in this case by induction on r.

- If $r = x$ the result is obvious.
- If $r = M \in \mathit{Dom}(\theta)$ then $\theta(M)$ is smaller than $s = \theta(l)$, and thus by the first i.h. $\#(\theta(M))$ is uniquely defined. This gives $\#(s) = \#(\theta(M))$ uniquely defined.
- If $r = f(t_1, \ldots, t_n)$, $r = a\langle t_1, \ldots, t_n\rangle$ or $r = \mu p.t$ the result holds by i.h. on r.
- If $r = t_1\{p \texttt{ by } t_2\}$, then by i.h. on r, $\#(\theta)(t_2)$ is uniquely defined. Since θ is deterministic, then $\rho = \{\!\{\#(\theta)(p) \texttt{ by } \#(\theta)(t_2)\}\!\}$ is uniquely defined by Remark 2. By Lemma 7 ρ is deterministic and thus so is $\theta \sqcup \rho$. We conclude that $\#(\theta)(r) = \#(\theta \sqcup \rho)(t_1)$ is uniquely defined by the second i.h.

We remark that the previous Lemma holds even when s is not a deterministic term. As an example let us take the classical λ-calculus as $\mathcal{R}$. Now let us take $s = \mathit{app}(\lambda a\langle x, y\rangle.a\langle 0, s(0)\rangle, 0)$ we then have $\#(s) = s$ even if s is a instance of the Beta-rule because this instance is not a deterministic one. And thus in fact s is a deterministic normal form for $\mathcal{R}$.

Definition 32 (Minimal Rigid Positions). *Given an orthogonal ERSP $\mathcal{R}$, the set of minimal rigid positions of a term s, denoted $\mathcal{MRP}(s)$, is defined as follows:*

- *If s is a redex $\theta(l)$, then $\mathcal{MRP}(s) = \mathcal{RP}(l)$.*
- *If s is not a redex, that is s is a variable, or $s = \mathcal{G}(s_1, \ldots, s_m)$ for $\mathcal{G}$ a variable, a case, a function or a binder symbol, then $\mathcal{MRP}(s) = \{\epsilon\} \cup \{i.\mathcal{MRP}(s_i) \mid i = 1, \ldots, m\}$.*

We remark that this definition needs the system $\mathcal{R}$ to be orthogonal to guarantee that $\mathcal{MRP}(s)$ is uniquely defined.

Lemma 15. *Let $\mathcal{R}$ be an orthogonal ERSP. Let $\ggg \in \{\stackrel{a}{\ggg}, \stackrel{c}{\ggg}, \stackrel{d}{\ggg}\}$. If $s \ggg s'$ and for every $p \in \mathcal{MRP}(s)$ we have that $s|_p$ is not a redex, then for every $o = s|_q$ with $q \in \mathcal{MRP}(s)$, there is $o' = s'|_q$ such that $o \ggg o'$.*

Proof. By induction on s. Let $s \ggg s'$ verifying the hypothesis. Suppose that s is a redex. Then by Definition 32 $\epsilon \in \mathcal{MRP}(s)$ and by hypothesis $s|_\epsilon = s$ is not a redex which yields a contradiction. So s is not a redex and thus by Remark 3 we have $s = \mathcal{G}(s_1, \ldots, s_m)$ and $s' = \mathcal{F}(s'_1, \ldots, s'_m)$, with $s_i \ggg s'_i$ for $i = 1, \ldots, m$. By i.h. for every $q \in \mathcal{MRP}(s_i)$, whenever $o = s_i|_q$, then there is a subterm $o' = s'_i|_q$ such that $o \ggg o'$. We have also $s|_\epsilon = s \ggg s' = s'|_\epsilon$ for $\epsilon \in \mathcal{MRP}(s)$. We can then conclude that the statement holds.

Lemma 16. *Let $\mathcal{R}$ be an orthogonal ERSP, let $\ggg \in \{\stackrel{a}{\ggg}, \stackrel{c}{\ggg}, \stackrel{d}{\ggg}\}$ and let $s \ggg s'$. If $s = \mathcal{G}(s_1, \ldots, s_m)$ and $s' = \mathcal{G}(s'_1, \ldots, s'_m)$ (where $\mathcal{G}$ is a function, binder symbol, case or variable) and $s_i \ggg s'_i$, then if s is a redex $\theta(l)$, s' is a also redex $\rho(l)$, for some ρ such that $\theta \ggg \rho$.*

Proof. By induction on s.

- If s is a variable the property vacuously holds since s cannot be a redex.
- If $\mathcal{G} = f$, then $s = \theta(l)$ implies that l is of the form $f(t_1, \ldots, t_m)$, with $\theta(t_i) = s_i$. Since the system is left-linear, we know that every term variable M of l occurs exactly once in it, and thus for every of such usual term occurring at position p, we define $\rho_v(M) = s'|_p$. To see that $s'|_p$ is well-defined let us consider the two following cases:
  - If $p \in \{1, \ldots, m\}$ (that is $t_p = M$), then $\theta(t_p) = s_p$ and $\rho(t_p) = s'_p$ so that $\theta(t_p) \ggg \rho(t_p)$ holds by hypothesis.
  - Otherwise, we know by orthogonality of $\mathcal{R}$ that there are no overlapping redexes in s so that for every $q \in \mathcal{MRP}(s)$ we have that $s|_q$ is not a redex. By Lemma 15 any rigid position p of l (that is any position p of $\mathcal{MRP}(s)$) is also a position of s' and $s|_p \ggg s'|_p$. Thus, $\rho_v(y) = s'|_p$ is well-defined.
- If $\mathcal{G}$ is a binder, the proof proceeds as before.
- If $\mathcal{G}$ is a case, then s cannot be a redex by definition of rewrite rule.

Lemma 17. *Let $\mathcal{R}$ be an l-constructor orthogonal ERSP. If $s \overset{c,d}{\ggg} s'$, then $s' \overset{c,d}{\ggg} \#(s)$, where $\overset{c,d}{\ggg} = \overset{c}{\ggg} \cap \overset{d}{\ggg}$.*

Proof. By induction on the definition of $s \overset{c,d}{\ggg} s'$.

- If $s = x$, then $s' = x$ and $\#(x) = x$. Thus the property obviously holds.
- If s is a deterministic instance $\theta(l)$ of the left-hand side l of some rule (l, r), then we have to show that $s' \overset{c,d}{\ggg} \#(s) = \#(\theta)(r)$. By Lemma 16 s' is also a redex $\rho(l)$, for some $\rho = (\rho_m, \rho_v)$, where $\rho_m = \theta_m$ and $\theta_t(x) \overset{c,d}{\ggg} \rho_t(x)$ for any usual variable x. We have then to show that $\rho(l) \overset{c,d}{\ggg} \#(\theta)(r)$. Since $\theta_v(x)$ is smaller than s we can apply the i.h. and we obtain that $\rho_v(x) \overset{c,d}{\ggg} \#(\theta_v(x))$. Thus we can conclude that $\rho(l) \overset{c,d}{\ggg} \#(\theta)(r)$ by Definition 30.
- All the other cases hold by the i.h.

Corollary 4. *If $\mathcal{R}$ is an l-constructor orthogonal ERSP, then the relation $\overset{c,d}{\ggg} = \overset{c}{\ggg} \cap \overset{d}{\ggg}$ is confluent on deterministic constructor acceptable terms.*

Proof. It is well-known that a relation having the diamond property is confluent [2] so that it is sufficient to show that $\ggg$ has the diamond property. Let us consider $s \ggg s'$ and $s \ggg s''$. By Lemma 17 we can close the diagram with $s' \ggg \#(s)$ and $s'' \ggg \#(s)$.

We now conclude with the main result of this section, namely,

Theorem 3. *Let $\mathcal{R}$ be an l-constructor orthogonal ERSP. The relation $\xrightarrow{c,d}_{\mathcal{R}}$ is confluent on acceptable constructor deterministic terms.*

Proof. By Corollaries 3 and 4.

6 Conclusion and further work

We have introduced a new *Higher-Order* formalism, called ERSP, in which the abstraction operation is not only allowed on variables but also on more complex patterns. This formalism can be seen as an extension of ERS [24] and SERS [5] to the case of patterns and an extension of [23] to the case of non functional rewriting rules. Many simple notions in the mentioned previous works do not trivially extend in our case: on one hand the complexity of ERSP does not only appear at the level of metaterms but also at the level of terms, on the other hand, binders are not always so simple as in the case of λ-calculus. We carefully extend all the expected notions of rewriting to our framework, namely, terms, metaterms, rewrite rules, substitutions, reduction, etc. The resulting formalism is able to modelize pattern-matching function/proof definitions.

The more technical part of this work is the identification of a class of ERSP which can be proved to be confluent on an appropriate set of terms. Our confluence result gives in particular a confluence result for SERS.

As mentioned in the introduction, ERSP and ρ-calculus [11] are closely related. The main difference between both formalisms lies in the class of syntactic patterns which are considered: our approach is mainly driven by the set of patterns appearing in functional languages and proof theorems, namely, algebraic patterns with choice constructors, while their approach includes other higher-order patterns (rules in their formalism) not really used in implementation of programming languages.

Many future directions remain to be explored. The first one consists in the definition of implementation languages given by "explicit" versions of this formalism, where both pattern matching and substitution operators are integrated to the base syntax. This would result in generalizations of calculi defined in [8, 14]. Also, a formal comparison between ERSP and ρ-calculus must be done. In particular, it would be interesting to know if every system in the ρ-calculus can be expressed in a ERSP system and vice-versa. Another interesting question concerns confluence in both formalism: confluent ρ-systems are characterized via a special *reduction strategy* while confluence is ensured in our case by syntactic restrictions.

Typing is another feature which remains as further work. It is however interesting to remark that the pioneer work on pattern calculi [23] which inspired the definition of ERSP was built, via the Curry-Howard isomorphism, on a computational interpretation of Gentzen sequent calculus for intuitionistic minimal logic. As a consequence, each ERSP pattern constructor comes from the interpretation of some *left* logical rule of Gentzen calculus. It is nevertheless less evident to associate a Curry-Howard style interpretation to the entire ERSP syntax.

Last but not least, strong normalization of ERSP has to be deeply studied. Indeed, proof techniques to guarantee termination of higher-order formalisms are not straightforward [4, 22, 37] and they do not extend immediately to our case.

References

1. The Alfa proof editor, `http://www.cs.chalmers.se/~hallgren/Alfa/`.
2. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.
3. H. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1984. Revised Edition.
4. F. Blanqui, J.-P. Jouannaud, and M. Okada. Inductive-Data-Type Systems. *Theoretical Computer Science*, 277, 2001.
5. E. Bonelli, D. Kesner, and A. Ríos. A de Bruijn notation for higher-order rewriting. In L. Bachmair (ed.), *11th International Conference on Rewriting Techniques and Applications*, *LNCS* 1833. Springer-Verlag, 2000.
6. R. Burstall, D. MacQueen, and D. Sanella. Hope: An experimental applicative language. In *Proceedings of the LISP Conference*, Stanford University, Computer Science Department, 1980.
7. The Caml language, `http://caml.inria.fr/`.
8. S. Cerrito and D. Kesner. Pattern matching as cut elimination. *Theoretical Computer Science*. To appear.

9. S. Cerrito and D. Kesner. Pattern matching as cut elimination. In G. Longo (ed.), *14th LICS*. IEEE Computer Society Press, 1999.
10. H. Cirstea. *Calcul de réécriture : fondements et applications*. Thèse de doctorat, Université Henri Poincaré - Nancy 1, 2000.
11. H. Cirstea and C. Kirchner. ρ-calculus, the rewriting calculus. In *5th International Workshop on Constraints in Computational Logics*, 1998.
12. The Coq Proof Assistant, `http://coq.inria.fr/`.
13. The ELAN system, `http://elan.loria.fr/`.
14. J. Forest. A weak calculus with explicit operators for pattern matching and substitution. In S. Tison (ed.), *13th International Conference on Rewriting Techniques and Applications*, *LNCS* 2378. Springer-Verlag, 2002.
15. J. Forest and D. Kesner. Expression Reduction Systems with Patterns, 2003. Available on `http://www.pps.jussieu.fr/~kesner/papers/`.
16. C. Kircher and G. Faure. Exceptions in the rewriting calculus. In S. Tison (ed.), *13th International Conference on Rewriting Techniques and Applications*, *LNCS* 2378. Springer-Verlag, 2002.
17. J. Goguen, J. Thatcher, and E. Wagner. An initial algebra approach to the specification, correctness and implementation of abstract data types. In *Current Trends in Programming Methodology* 4. Prentice Hall, 1978.
18. C. Kirchner, H. Cirstea and L. Liquori. The Rho Cube. In F. Honsell and M. Miculan (eds.), *Foundations of Software Science and Computation Structures*, *LNCS* 2030. Springer-Verlag, 2001.
19. P. Hudak, S. Peyton-Jones, and P. Wadler (eds). Report on the programming language haskell, a non-strict, purely functional language (version 1.2). Sigplan Notices, 1992.
20. G. Huet and D. Oppen. Equations and rewrite rules: A survey. In R.V.Book (ed.), *Formal Language Theory: Perspectives and Open Problems*. Academic Press, 1980.
21. The Isabelle theorem prover, `http://isabelle.in.tum.de/`.
22. J.-P. Jouannaud and A. Rubio. The higher-order recursive path ordering. In G. Longo (ed.), *14th LICS*. IEEE Computer Society Press, 1999.
23. D. Kesner, L. Puel, and V. Tannen. A Typed Pattern Calculus. *Information and Computation*, 124(1), 1996.
24. Z. Khasidashvili. Expression reduction systems. In *Proceedings of IN Vekua Institute of Applied Mathematics*, volume 36, Tbilisi, 1990.
25. J.-W. Klop. *Combinatory Reduction Systems*, volume 127 of *Mathematical Centre Tracts*. CWI, Amsterdam, 1980. PhD Thesis.
26. J.-W. Klop. *Term rewriting systems*, volume 2. Oxford University Press, 1992.
27. J.-W. Klop, V. van Oostrom, and F. van Raamsdonk. Combinatory reduction systems: introduction and survey. *Theoretical Computer Science* 121, 1993.
28. The MAUDE System, `http://maude.cs.uiuc.edu/`.
29. R. Milner, M. Tofte, and R. Harper. *The definition of Standard ML*. MIT Press, 1990.
30. T. Nipkow. Higher-order critical pairs. In *6th LICS*. IEEE Computer Society Press, 1991.
31. The Objective Caml language, `http://caml.inria.fr/`.
32. The PVS system, `http://pvs.csl.sri.com/`.
33. M. Takahashi. Parallel Reductions in lambda-Calculus. *Journal of Symbolic Computation*, 7(2), 1989.
34. D. Turner. Miranda: A non-strict functional language with polymorphic types. In J. P. Jouannaud (ed.), *Functional Programming Languages and Computer Architecture*, *LNCS* 201. Springer-Verlag, 1985.
35. V. van Oostrom and F. van Raamsdonk. Weak orthogonality implies confluence: the higher-order case. In A. Nerode and Y. Matiyasevich (eds.), *Proceedings of the Third International Symposium on Logical Foundations of Computer Science*, *LNCS* 813. Springer-Verlag, 1994.
36. V. van Oostrom and F. van Raamsdonk. Weak orthogonality implies confluence: the higher-order case. In A. Nerode and Y. Matiyasevich (eds.), *Proceedings of the Third International Symposium on Logical Foundations of Computer Science*, *LNCS* 813. Springer-Verlag, 1994.

37. F. van Raamsdonk. On termination of higher-order rewriting. In A. Middeldorp (ed.), *12th International Conference on Rewriting Techniques and Applications*, *LNCS* 2051. Springer-Verlag, 2001.
38. D. Wolfram. *The Clausal Theory of Types*, volume 21 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1993.