Expression Reduction Systems with Patterns

Julien Forest (forest@sophia.inria.fr)
INRIA Sophia-Antipolis, France

Delia Kesner (kesner@pps.jussieu.fr)
PPS (CNRS UMR 7126), Université Paris 7, France

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 with metavariables. These patterns are built by combining standard algebraic patterns with choice constructors denoting cases. In other words, the non deterministic choice between different rewrite 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.
\[ \Rightarrow c,d \]

\[ \Rightarrow^* c,d \]

\[ \rightarrow p \]

\[ \Rightarrow p \]

\[ \rightarrow p \]

\[ \Rightarrow p \]

\[ \Rightarrow p \]

\[ \Rightarrow p \]

\[ \Rightarrow p \]

\[ \Rightarrow i \]

\[ \Rightarrow p \]

\[ \Rightarrow i \]

\[ \Rightarrow p \]

\[ \Rightarrow i \]

\[ \Rightarrow p \]

\[ \Rightarrow d \]

\[ \Rightarrow p \]

\[ \Rightarrow d \]

\[ \Rightarrow p \]

\[ \Rightarrow i,d \]

\[ \Rightarrow p \]

\[ \Rightarrow i,d \]

\[ \Rightarrow p \]

\[ \Rightarrow c,d \]

\[ \Rightarrow p \]

\[ \Rightarrow c,d \]

\[ \Rightarrow p \]

\[ \Rightarrow i,c,d \]

\[ \Rightarrow p \]

Higher-order rewrite systems are able to combine formalisms coming from proof theory, such as \(\lambda\)-calculus, with formalisms arising in algebraic specifications, such as first-order rewrite systems (Goguen et al., 1978; Huet and Oppen, 1980; Klop, 1992). 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 encoded by higher-order rewrite systems.

Many higher-order rewrite systems exist in the literature (Khasidashvili, 1990; Nipkow, 1991; van Oostrom and van Raamsdonk, 1994; Wolfram, 1993; Bonelli et al., 2000; Aczel, 1978) starting with the seminal work by J-W. Klop (Klop, 1980). The theory of higher-order rewriting is considerably more involved than that of first-order rewriting; many articles were devoted 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 (Hope, OCaml, Haskell, ML, Miranda) and proof assistants (Alfa, Coq, Elan, Isabelle, Maude, PVS) admit definitions by cases via pattern-matching mechanisms. Thus, a natural extension of higher-order rewriting consists in the use of binders for patterns so that a projection function like \(\lambda(x,y).x\) would be also acceptable.

The Pattern-Matching Calculus (Kesner et al., 1996), proposed as a theoretical framework to study pattern-matching in a pure functional paradigm, admits precisely this kind of binding mechanisms. Its evaluation process is given by the following generalization of the standard \(\beta\)-rule to the case of patterns:

\[
(\beta_{PM}) \quad \text{app}(\lambda X. M, N) \rightarrow M\{X \text{ by } N\}
\]

where \(X\) denotes a pattern and \(\{X \text{ by } N\}\) denotes a substitution resulting from the application of the pattern-matching operation to the pattern \(X\) and to the term \(N\).

This calculus was later extended with explicit operators (Cerrito and Kesner, 1999; Cerrito and Kesner, 2004; Forest, 2002); weak reduction was widely studied in (Forest, 2002). Other languages allowing abstractions on patterns (without metavariables) were studied in (van Oost-

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 and metavariables are used not only to construct metaterms but also to construct metapatterns. Our calculus constitutes an extension of ERSs (Khasidashvili, 1990) and SERSs (Bonelli et al., 2000) 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 by the following ERSP term:

\[ \lambda a(\langle \text{nil, cons}(x, t) \rangle).a(0, \text{plus}(1, \text{length}(t))) \]

where \( \lambda \) is a binder, \( a \) is a special variable used to identify the choice over patterns \( \text{nil} \) and \( \text{cons}(x, t) \) with the set of their corresponding continuations \( 0 \) and \( \text{plus}(1, \text{length}(t)) \).

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 (Nipkow, 1991; van Oostrom and van Raamsdonk, 1994). Much more, our confluence result turns out to give in particular a confluence result for SERSs.

It is worth noticing that the presence of pattern metavariables in ERSP rewrite rules makes the confluence proof more subtle than those of pattern calculi such as (van Oostrom, 1990; Cirstea and Kirchner, 1998; Kahl, 2003; Jay and Kesner, 2006). Indeed, in calculi without pattern metavariables the orthogonality property of the rewrite system usually suffices to ensure confluence of the reduction relation. However, if quantification over patterns is allowed, then orthogonality and well-formedness of the rewrite rules is not enough to guarantee confluence of the reduction relation. Indeed, consider the rule \( \beta_{PM} \) previously introduced together with a second rewrite rule of the form \( f \rightarrow g \). Then the term \( \text{app}(\lambda f.h, f) \) reduces to both \( h \) and \( \text{app}(\lambda f.h, g) \), which are not joinable terms. The problem comes from the fact that the pattern \( \mathbf{X} \) in
the left-hand side of the rule $\beta_{PM}$ has been instantiated with a term which is not a constructor (as it can be further reduced).

As a consequence, one needs to propagate orthogonality and well-formedness to all the metasubstitutions and substitutions generated during the reduction process, thus resulting in a class of systems which is confluent for a particular reasonable class of terms.

This paper is an extended version of (Forest and Kesner, 2003). It is organized as follows. Sections 1 and 2 introduce the basic ingredients of the syntactic formalism ERSP. In Section 3 we develop some examples in our framework. Section 4 is devoted to study a restriction of the class of ERSP so that confluence will follow (Section 5). We conclude and give many further research directions in Section 6.

1. Basic notions of the ERSP formalism

Besides all the constructors used to build metaterms, we use the following alphabets: a set $UV$ of usual variables denoted $x, y, z, \ldots$, a set $CV$ of choice variables denoted $a, b, c, \ldots$, a set $F$ of function symbols equipped with a fixed arity ($n \geq 0$), denoted $f, g, h, \ldots$, a set $B$ of binder symbols denoted $\lambda, \mu, \nu, \ldots$, a set $PV$ of pattern metavariables denoted $X, Y, \ldots$, a set $TV$ of term metavariables denoted $M, N, \ldots$. We assume all these sets to be denumerable and pairwise disjoint.

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

Metapatterns $(p)$ are generated by the grammar:

$$
p ::= x \quad \text{usual variable} \\
| \quad X \quad \text{pattern metavariable} \\
| f(p, \ldots, p) \quad \text{algebraic} \\
| a(p, \ldots, p) \quad \text{choice} \\
| \quad @(p, \ldots, p) \quad \text{layered} \\
| \quad \_ \quad \text{wildcard}
$$

The constructor $@()$ is varyadic, i.e. it has no fixed arity. It can be understood as the natural generalization of the `as` construction in OCaml. The constructor $a(\_)$ is also varyadic, but with an arity strictly greater than 0 (even if $a(t)$ and $t$ will be semantically identified).

Metaterms $(t)$ are generated by the grammar:
We assume that whenever \( a \) appears inside a metaterm \( t \), then all its occurrences have the same arity: thus, the term \( \mu a(x).a(x, y) \) is not allowed. The symbol \( \{ \text{by} \} \) is called the pattern-matching constructor.

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 \( \mathcal{V} \text{ar}(p) \) the set of all the variables (usual and choice) appearing in a metapattern \( p \). We denote by \( \mathcal{M}\mathcal{V}(t) \) the set of all the pattern and term metavariables appearing in \( t \).

**DEFINITION 1.1.** 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 following equations

\[
\begin{align*}
\text{fib}(0) &= 0 \\
\text{fib}(1) &= 1 \\
\text{fib}(x+2) &= \text{fib}(x) + \text{fib}(x+1)
\end{align*}
\]

Using a choice variable \( a \) of arity 3 to encode the three different choices given by the previous specification, an encoding of \( \text{fib} \) in our syntax could be given by:

\[
\lambda a\langle 0, s(0), s(s(x)) \rangle, a\langle 0, s(0), \text{plus}(\text{fib}(x), \text{fib}(s(x))) \rangle
\]

where \( \lambda \) is a binder, and natural numbers are encoded by 0, \( s(0) \), \( s(s(0)) \), \ldots (even if sometimes they may be denoted by 0, 1, 2, \ldots ).

We will come back to this example later on.

A position is a word over the alphabet \( \mathbb{N} \); we use \( \epsilon \) to denote the empty word. The set of positions of a metaterm \( t \), denoted \( \mathcal{P}\mathcal{O}\mathcal{S}(t) \), is defined as usual (Baader and Nipkow, 1998) except for the term.
s\{p \text{ by } u\} for which we define $1.q \in \mathcal{POS}(s\{p \text{ by } u\})$ if $q \in \mathcal{POS}(s)$ and $2.q \in \mathcal{POS}(s\{p \text{ by } u\})$ if $q \in \mathcal{POS}(u)$ (see also (Bonelli et al., 2000)). The justification of this case comes from the fact that, when reasoning about positions, the metaterm $s\{p \text{ by } u\}$ can be seen as “$\text{app}(\mu p.s,u)$”, where $\text{app}$ is a function symbol and $\mu$ is a binder. The submetaterm of $t$ at position $q$ is written as $t|q$. When $t|q = u$, we will say that $q$ is an occurrence of $u$ in $t$. We will use the notation $t[u']_{q}$ to denote the replacement of $t|q$ by $u'$ in $t$.

The following notion is used to describe the set of variables and metavariables appearing along a “binding” path.

**DEFINITION 1.2. (Parameter path).** Given a metaterm $s$ and $q \in \mathcal{POS}(s)$, we define the parameter path of $s$ at position $q$, written $PP(s,q)$, as the following subset of variables and metavariables of $s$:

\[
PP(s,\epsilon) = \emptyset \\
PP(f(s_1,\ldots,s_n),i.q) = PP(s_i,q), \text{ for } i \in \{1,\ldots,n\} \\
PP(a(s_1,\ldots,s_n),i.q) = PP(s_i,q), \text{ for } i \in \{1,\ldots,n\} \\
PP(\mu p.s,1.q) = \text{Var}(p) \cup \text{MV}(p) \cup PP(s,q) \\
PP(u\{p \text{ by } v\},1.1.q) = \text{Var}(p) \cup \text{MV}(p) \cup PP(u,q) \\
PP(u\{p \text{ by } v\},2.q) = PP(v,q)
\]

As an example, if $t = M\{g(X,x) \text{ by } \mu a(Y,s(Y)).N\}$, then we have $PP(t,2) = \emptyset$, $PP(t,1.1) = \{X,x\}$, and $PP(t,2.1) = \{Y,a\}$.

We assume that different nested metapatterns appearing on the same path cannot share (meta)variables. Thus for example, $\mu X.\lambda X.M$ or $\lambda x.\mu x.M$ are not allowed but the metaterm $t$ given above is allowed since the two occurrences of $Y$ in $t$ are not nested. This is just a generalization of what is called “Barendregt’s convention on bound variables”.

**DEFINITION 1.3. (Free and bound variables).** The set of free variables of a metaterm is defined by induction as follows:

\[
\mathcal{FV}(x) = \{x\} \\
\mathcal{FV}(M) = \emptyset \\
\mathcal{FV}(f(t_1,\ldots,t_n)) = \mathcal{FV}(t_1) \cup \ldots \cup \mathcal{FV}(t_n) \\
\mathcal{FV}(a(t_1,\ldots,t_n)) = \{a\} \cup \mathcal{FV}(t_1) \cup \ldots \cup \mathcal{FV}(t_n) \\
\mathcal{FV}(\mu p.u) = \mathcal{FV}(u) \setminus \text{Var}(p) \\
\mathcal{FV}(t\{p \text{ by } u\}) = (\mathcal{FV}(t) \setminus \text{Var}(p)) \cup \mathcal{FV}(u)
\]

Accordingly, the set of bound variables of a metaterm is defined by induction as follows:
\[ \mathcal{BV}(x) = \emptyset \]
\[ \mathcal{BV}(M) = \emptyset \]
\[ \mathcal{BV}(f(t_1, \ldots, t_n)) = \mathcal{BV}(t_1) \cup \ldots \cup \mathcal{BV}(t_n) \]
\[ \mathcal{BV}(a(t_1, \ldots, t_n)) = \mathcal{BV}(t_1) \cup \ldots \cup \mathcal{BV}(t_n) \]
\[ \mathcal{BV}(\mu p.u) = \mathcal{BV}(u) \cup \mathcal{Var}(p) \]
\[ \mathcal{BV}(t \{p \text{ by } u\}) = \mathcal{BV}(t) \cup \mathcal{Var}(p) \cup \mathcal{BV}(u) \]

We work modulo \(\alpha\)-conversion on preterms, so that renaming of bound variables can be used to avoid clashes. Thus for example:

\[ \mu a \langle x, y, z \rangle. a \langle x, x, v \rangle \alpha \mu b \langle x', y', z' \rangle. b \langle x', x', v \rangle \]

As a consequence, and without any loss of generality, we will assume the sets of free and bound variables of preterms to be disjoint.

**DEFINITION 1.4. (Well-formed metaterm).** A metaterm \(t\) is well-formed iff \(t\) has no free occurrences of choice/usual variables.

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

**DEFINITION 1.5. (Contexts).** Contexts are terms with one (and only one) occurrence containing a distinguished constant called a “hole” (and denoted \(\Box\)) in a (sub)term position. Thus \(\mu X.\Box\) is a context but \(\mu \Box.y\) is not. We usually denote a generic context by \(C[\_]\).

**DEFINITION 1.6. (Replacement and substitution).**

- A replacement (or metasubstitution) is a denumerable set of pairs of the form \(X \triangleright p\) and \(M \triangleright t\) where \(p\) is a pattern and \(t\) is a term. Replacements will be denoted by \(\rho, \delta, \sigma, \theta, \ldots\).

- A substitution is a denumerable set of pairs of the form \(x \triangleright t\) and \(a \triangleright i\), where \(t\) is a term and \(i\) is a natural number. Substitutions will be denoted by \(\rho, \delta, \sigma, \theta, \ldots\).

When no special distinction is needed between replacements and substitutions, we will use the symbols \(\hat{\rho}, \hat{\delta}, \hat{\sigma}, \ldots\). We denote by \(\text{id}\) the empty replacement/substitution.

The domain of a replacement (resp. substitution) \(\hat{\sigma}\) is defined as \(\text{Dom}(\hat{\sigma}) = \{\hat{x} \mid \hat{x} \triangleright o \in \sigma \text{ and } \hat{x} \neq o\}\). When \(\hat{x} \in \text{Dom}(\hat{\sigma})\) we write \(\hat{\sigma} \hat{x}\) to denote the object \(o\) such that \(\hat{x} \triangleright o \in \hat{\sigma}\). The codomain of \(\hat{\sigma}\) is given by \(\text{Codom}(\hat{\sigma}) = \bigcup_{\hat{x} \in \text{Dom}(\hat{\sigma})} \mathcal{FV}(\hat{\sigma} \hat{x})\).
The union of two replacements/substitutions \( \hat{\theta}_1 \) and \( \hat{\theta}_2 \) is denoted by \( \hat{\theta}_1 \sqcup \hat{\theta}_2 \). This union is only defined if \( \hat{\theta}_1 \hat{x} = \hat{\theta}_2 \hat{x} \) for every variable \( \hat{x} \in \text{Dom}(\hat{\theta}_1) \cap \text{Dom}(\hat{\theta}_2) \).

We are now ready to define the notion of pattern-matching. This operation is not defined as a function from patterns and terms to substitutions but from patterns and terms to sets of substitutions. We will see later how to force this set to be a singleton.

**DEFINITION 1.7.** (Pattern-matching). Pattern-matching is a partial operation associated to patterns and terms and yielding sets of substitutions given as follows:

- \( id \in \{ \langle \_ \rangle \; \text{by} \; t \} \)
- \( \{x \triangleright t\} \subseteq \{x \; \text{by} \; t\} \)
- \( \hat{\theta}_1 \sqcup \ldots \sqcup \hat{\theta}_n \in \{ \langle p_1, \ldots, p_n \rangle \; \text{by} \; t \} \) if \( \forall i \hat{\theta}_i \in \{ p_i \; \text{by} \; t \} \)
- \( \hat{\theta}_1 \sqcup \ldots \sqcup \hat{\theta}_n \in \{ f(p_1 \ldots p_n) \; \text{by} \; f(t_1 \ldots t_n) \} \) if \( \forall i \hat{\theta}_i \in \{ p_i \; \text{by} \; t_i \} \)
- \( \{a \triangleright i\} \sqcup \hat{\theta}_i \in \{a \langle p_1 \ldots p_n \rangle \; \text{by} \; t\} \) if \( 1 \leq i \leq n \) & \( \hat{\theta}_i \in \{ p_i \; \text{by} \; t \} \)

We remark that in the last three cases the result of \( \{ p \; \text{by} \; t \} \) is defined only if \( \sqcup \) is defined. When \( \{ p \; \text{by} \; t \} \) is a singleton we will make an abuse of notation by writing \( \{ p \; \text{by} \; t \} \) to denote the only element of this set.

Remark also that the last case of the previous definition is the only one yielding multiple solutions. Thus for example, the pattern-matching \( \{a(0, x) \; \text{by} \; 0\} \) has two solutions: \( \{a \triangleright 1\} \) and \( \{a \triangleright 2, x \triangleright 0\} \). This comes from the fact that the pattern \( a(0, x) \) contains two ”overlapping” subpatterns 0 and \( x \).

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

In CRSs (Klop, 1980; Klop et al., 1993) 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 \( \beta \)-rule has to be written as \( \text{app}(\lambda x. M(x), N) \rightarrow M(N) \) which does not correspond to the traditional way to express the \( \beta \)-rule.

In ERSs (Khasidashvili, 1990) metasubstitution is used to express the \( \beta \)-rule in a traditional way as \( \text{app}(\lambda x. M, N) \rightarrow M\{x/N\} \). The instantiation of the metavariable \( M \) may or may not capture the variable \( x \). However, we cannot assume a naive \( \alpha \)-conversion rule 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 \( \alpha \)-equivalent terms \( \lambda x.x \neq_\alpha \lambda y.x \).

In order to properly handle \( \alpha \)-conversion in the whole formalism we will simply extend to metaterms the equality \( =_\alpha \) previously defined on preterms by adding \( t =_\alpha t[u]_q \), where \( q \in \mathcal{POS}(t) \) and \( t|_q \) and
are \(\alpha\)-equivalent preterms. In this way the congruence on preterms generated by \(\alpha\)-conversion is naturally and safely lifted to metaterms. Remark in particular that the equality \(\lambda x. M =_\alpha \lambda y. M\) does not hold within this notion of \(\alpha\)-conversion. A consequence of this notion will be that \(\alpha\)-conversion is stable by instantiation: if two metaterms \(t\) and \(t'\) are \(\alpha\)-equivalent, then instantiating them will yield two \(\alpha\)-equivalent terms. It is worth noticing that even if \(\alpha\)-conversion is only defined on preterms (as done in (Forest and Kesner, 2003)) and not on metaterms, this stability property is still true.

Now, let us come back to the instantiation of rules in higher-order rewriting which is usually decomposed in two different steps: a first-order replacement which is performed in order to give a concrete value to all the metavariables, so that capture of variables can be provoked, and a higher-order substitution (modulo \(\alpha\)-conversion) which is the one “computing” the redexes created by the first step.

**DEFINITION 1.8. (Applying a substitution).** The application of a substitution \(\theta\) to a preterm \(t\) (or instantiation of \(t\) by \(\theta\)) yields a set of terms, written \(\theta(t)\), which is computed as a higher-order substitution (modulo \(\alpha\)-conversion) as follows:

\[
\begin{align*}
\theta x & \in \theta(x) & \text{if } x \in \text{Dom}(\theta) \\
x & \in \theta(x) & \text{if } x \notin \text{Dom}(\theta) \\
\mu p.t' & \in \theta(\mu pt) & \text{if } t' \in \theta(t) \text{ and no capture of variables holds} \\
f(t_1', \ldots, t_n') & \in \theta(f(t_1, \ldots, t_n)) & \text{if } t_i' \in \theta(t_i) \\
t_i' & \in \theta(a(t_1, \ldots, t_n)) & \text{if } \theta a = i \text{ and } t_i' \in \theta(t_i) \\
a(t_1', \ldots, t_n') & \in \theta(a(t_1, \ldots, t_n)) & \text{if } t_i' \in \theta(t_i) \text{ and } a \notin \text{Dom}(\theta) \\
t' & \in \theta(t\{p \text{ by } u\}) & \text{if } u' \in \theta(u) \text{ and } \theta' \in \{p \text{ by } u'\}, \text{ and } \\
& & \text{if } t' \in (\theta' \cup \theta')(t) \text{ and no capture of variables holds}
\end{align*}
\]

**DEFINITION 1.9. (Applying a replacement).** The first-order replacement \(\theta\) of a metaterm \(t\) is written \(\theta(t)\). The application of a replacement \(\theta\) to a metaterm \(t\) (or instantiation of \(t\) by \(\theta\)) yields a set of terms, written \(\text{THETA}(t)\), given by \(\text{id}(\theta(t))\), where \(\text{id}\) is the empty substitution.

Remark that whenever a metaterm \(t\) has no pattern-matching constructor, then, if defined, \(\text{THETA}(t) = \theta(t)\) is at most a singleton so that we will identify in this case the singleton with its unique element.

Let us take \(\theta = \{X \triangleright a(x, f(z, y)), M \triangleright a(g(x, x), z), N \triangleright f(x, x)\}\) as an example. In order to compute \(\text{THETA}(M\{X \text{ by } N\})\) we first compute
the first-order replacement

\[ t = \text{theta}(M\{X \text{ by } N\}) = a\langle g(x, x), z\rangle\{a\langle x, f(z, y) \text{ by } f(x, x) \rangle \} \]

Now, since \( \alpha \)-conversion is allowed on preterms, we obtain

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

The computation of \( \{a\langle x', f(z, y) \text{ by } f(x, x) \rangle \} \) 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 application of \( \text{theta} \) to \( M\{X \text{ by } N\} \) gives a set

\[ \text{THETA}(M\{X \text{ by } N\}) = \text{id}(\text{theta}(M\{X \text{ by } N\})) = \text{id}(t) =_\alpha \text{id}(t') = \{g(f(x, x), f(x, x)), x\} \]

2. Rewrite rules and reduction relation

This section introduces the syntax used to specify rewrite rules in the ERS\( P \) formalism as well as the reduction relation associated to them.

**DEFINITION 2.1.** An Expression Reduction System with Patterns (ERS\( P \)) is a set of rewrite rules of the form \( l \rightarrow r \) (written also \( (l, r) \)) such that:

- \( l \) and \( r \) are well-formed metaterms,
- the first symbol (called head symbol) in \( l \) is in \( F \cup B \),
- \( \text{MV}(r) \subseteq \text{MV}(l) \),
- \( l \) contains no occurrence of the pattern-matching constructor.

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

Before defining the notion of reduction relation we remark that given a rewrite rule \( l \rightarrow r \), \( \text{THETA}(l) \) is a singleton for any replacement \( \text{theta} \).

**DEFINITION 2.2.** (Reduction relation). Let \( \mathcal{R} \) be an ERS\( P \). We say that \( s \) rewrites to \( t \), written \( s \rightarrow_\mathcal{R} t \) iff there exists a rule \( (l, r) \in \mathcal{R} \), a replacement \( \text{theta} \) and a context \( C \) such that \( s = C[\text{THETA}(l)] \) and \( t \in C[\text{THETA}(r)] \). This notion can also be defined by induction:

\[
\begin{align*}
(l, r) &\in \mathcal{R} \text{ and } t \in \text{THETA}(r) \quad &\text{THETA}(l) \rightarrow_\mathcal{R} t \quad &\text{t} \rightarrow_\mathcal{R} u \text{ and } C \text{ is a context} \quad &\text{C}[t] \rightarrow_\mathcal{R} \text{C}[u]
\end{align*}
\]
Remark that patterns cannot be reduced in this framework. As usual, we denote by $\rightarrow_R$ the reflexive and transitive closure of $\rightarrow_R$.

Unfortunately, this notion may introduce new free variables during reduction making the semantics of the formalism wrong: thus for example, take the term $t_0 = \text{app}(\mu a(f(x), g(y)).a(y, x), f(0))$ and consider the following reduction sequence:

$$
\begin{align*}
t_0 & \rightarrow^{\beta_{PM}} a\langle y, x \rangle_{\{a(f(x), g(y)) \text{ by } f(0)\}} = \\
a\langle y, x \rangle_{\{a \triangleright 1, x \triangleright 0\}} & = y
\end{align*}
$$

Then, we have $\mathcal{FV}(t_0) = \emptyset$ while $y$ is free in the reduct of $t_0$.

In order to avoid these pathological cases we introduce a notion of admissibility for terms and reductions in order to guarantee preservation of free variables. For that, let us start by defining subsets of free variables which remain free after substitutions on a choice variable.

**Definition 2.3.** (Localized free variables). Given $a \in \mathcal{CV}$, $i \geq 1$ and a preterm $t$, the set $\mathcal{FV}^i_a(t)$ of localized free variable of $t$ is defined modulo $\alpha$-conversion (so that we can assume that $a$ is not a bound variable of $t$) as follows:

$$
\begin{align*}
\mathcal{FV}^i_a(x) & = \{x\} \\
\mathcal{FV}^i_a(f(t_1, \ldots, t_n)) & = \mathcal{FV}^i_a(t_1) \cup \ldots \cup \mathcal{FV}^i_a(t_n) \\
\mathcal{FV}^i_a(a(t_1, \ldots, t_n)) & = \mathcal{FV}^i_a(t_i) \quad \text{if } i \leq n \\
\mathcal{FV}^i_a(b(t_1, \ldots, t_n)) & = \{b\} \cup \mathcal{FV}^i_a(t_1) \cup \ldots \cup \mathcal{FV}^i_a(t_n) \quad \text{if } a \neq b \\
\mathcal{FV}^i_a(\mu p.u) & = \mathcal{FV}^i_a(u) \setminus \text{Var}(p) \\
\mathcal{FV}^i_a(t\{p \text{ by } u\}) & = (\mathcal{FV}^i_a(t) \setminus \text{Var}(p)) \cup \mathcal{FV}^i_a(u)
\end{align*}
$$

Thus, for example, we have $\mathcal{FV}^i_a(b(x, y, z)) = \{b, x, y, z\}$ for any $i$, and $\mathcal{FV}^i_a(a\langle x, y, z \rangle) = \{x\}$, and

$$
\mathcal{FV}^i_a(\mu a(x, y).a(f(x, z), u)) = \mathcal{FV}^i_a(\mu b(x, y).b(f(x, z), u)) = \{z, u\}
$$

Remark also that $\mathcal{FV}^i_a(t) = \mathcal{FV}(t)$ for every $i$ whenever the choice variable $a$ is not free in $t$.

**Definition 2.4.** (Acceptable preterms). Acceptability is the least relation on preterms such that:

- All the usual variables are acceptable.
- 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, \( p \) is a pattern, and for all \( a(p_1, \ldots, p_n) \in p \), for all \( i \in 1 \ldots n \), and for all \( j \neq i \) we have \((\text{FV}_a(t) \setminus \text{Var}(p_j)) \cap \text{Var}(p_i) = \emptyset\), then \( \mu p . t \) is an acceptable term.

If \( \mu p . t \) and \( u \) are acceptable, then \( t \{ p \text{ by } u \} \) is acceptable.

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 both acceptable while \( \mu a \langle x, y \rangle . b \langle x, y \rangle \) is not since the variable \( y \) belongs to both \( \text{Var}(y) \) and \( \text{FV}_a(b \langle x, y \rangle) \setminus \text{Var}(x) = \{ y, b \} \). The term \( \mu a \langle x, y \rangle . a \langle y, x \rangle \) is not acceptable as well.

The notion of acceptability is not closed by contexts: if a preterm \( t \) is acceptable, then \( C[t] \) is not necessarily acceptable. Indeed, the preterm \( a(x, y) \) is acceptable while \( \lambda a \langle y, x \rangle . a \langle x, y \rangle \) is not.

**Definition 2.5.** (Acceptable replacement/substitution). Let \( S \) be a set of term metavariables (resp. usual variables). A replacement (resp. substitution) \( \hat{\theta} \) is said to be acceptable w.r.t. \( S \) iff for every metavariable \( \hat{x} \in S \) the term \( \hat{\theta} \hat{x} \) is acceptable. A replacement (resp. substitution) \( \hat{\theta} \) is said to be acceptable if it is acceptable w.r.t. \( \text{Dom}(\hat{\theta}) \).

**Lemma 2.6.** Let \( t \) be an acceptable term and \( p \) be a pattern. Then any substitution \( \sigma \in \{ p \text{ by } t \} \) is acceptable.

**Proof.** By induction on \( p \).

- If \( p = \_ \) then, by definition of \( \{ \_ \text{ by } \_ \} \), we know that \( \sigma = \text{id} \) which is obviously acceptable.

- If \( p = x \) the result is straightforward by hypothesis.

- If \( p = f(p_1, \ldots, p_n) \), then, by definition of \( \{ p \text{ by } t \} \), we know that \( t = f(t_1, \ldots, t_n) \) and that for all \( i \) the set \( \{ p_i \text{ by } t_i \} \) is defined. By definition of acceptable term, we know that all the \( t_i \)'s are acceptable and thus by i.h. we have that every \( \{ p_i \text{ by } t_i \} \) only contains acceptable substitutions. The result is then straightforward since the union of acceptable substitutions, if defined, is an acceptable substitution.

- For the other cases we reason as in the previous case.

**Lemma 2.7.** Let \( t \) be a preterm and let \( \theta \) be an acceptable substitution w.r.t. \( \text{FV}(t) \). Then, the set \( \theta(t) \) only contains acceptable terms.

**Proof.** By induction on \( t \).
- If \( t = x \), then the result is obvious by hypothesis.

- If \( t = f(t_1, \ldots, t_n) \) or if \( t = a(t_1, \ldots, t_n) \) the result is obvious by i.h.

- If \( t = \mu p. t' \) then we assume \( \text{Codom}(\sigma) \cap \text{Var}(p) = \emptyset \) by \( \alpha \)-conversion, thus the result is straightforward by i.h.

- If \( t = t'\{p \text{ by } u\} \), then by \( \alpha \)-conversion, we can suppose that \( \text{Codom}(\sigma) \cap \text{Var}(p) = \emptyset \). By i.h., we then know that \( \sigma(u) \) only contains acceptable terms. So, by Lemma 2.6 we have that for any \( u' \) in \( \sigma(u) \) the set \( \{p \text{ by } u'\} \) only contains acceptable substitutions. Since the union of acceptable substitutions, if defined, is an acceptable substitution, then for any \( \theta' \in \{p \text{ by } u'\} \), \( \theta \sqcup \theta' \) is acceptable w.r.t. \( \text{FV}(t') \). We then conclude by i.h.

In order to guarantee that no free variable is “generated” during reduction the following notion is also necessary.

DEFINITION 2.8. (Path condition). Let \( M \) be a term metavariable in the metaterm \( t \). 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 \( \theta \) is said to have the path condition property for \( M \) in \( t \) iff:

\[
\forall 1 \leq i \leq n, \hat{x} \in \text{FV}(\theta(M)) \quad \text{or} \quad \forall 1 \leq i \leq n, \hat{x} \notin \text{THETA}(l_i)
\]

where the notation \( \text{THETA}(l) \) denotes the set \( \bigcup_{\hat{x} \in l} \text{FV}(\theta(\hat{x})) \).

This notion is extended to rewrite rules by saying that \( \theta \) has the path condition for \( M \) in \( (l, r) \) iff it has the path condition for \( M \) in \( \mapsto (l, r) \), where \( \mapsto \) is a fresh binary function symbol. This trick is 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 \( \eta \)-rule \( \lambda x.\text{app}(M, x) \rightarrow M \). Another rule in the same spirit but using patterns is \( \lambda f(X).M \rightarrow M \): the replacement \( \theta = \{X \triangleright x, M \triangleright x\} \) does not satisfy the path condition for \( M \) in this rule.

DEFINITION 2.9. (Admissible replacements). A replacement \( \theta \) is admissible for a metaterm \( t \) iff

- \( \theta \) contains only acceptable terms
A replacement $\theta$ is admissible for a rule $(l, r)$ iff $\theta$ is admissible for $l \rightarrow (l, r)$, where $\rightarrow$ is a fresh 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 $\text{Dom}(\theta)$.

The admissible reduction relation, denoted by $a \rightarrow$ or simply by $\rightarrow$ when no confusion is possible, is the one generated only by admissible replacements. In this case, even if the relation $\rightarrow_R$ is defined by using any kind of contexts, the reduction can only take place on acceptable subterms. From now on, we only consider admissible reductions.

As expected, the reduction relation enjoys good preservation properties.

**Lemma 2.10.** Let $l \rightarrow r$ be a rewrite rule and let $\theta$ be an admissible replacement for $l \rightarrow r$. Then $\forall s \in \Theta(r) \forall a \forall i FV_a(s) \subseteq FV_a(\Theta(l))$.

**Proof.** Let us take any $s \in \Theta(r)$ and let us suppose that $x \in FV_a(s)$. Two cases are possible:

1. If $x \notin FV(r)$, then $r$ is not well-formed leading to a contradiction.

2. If $x \notin FV(r)$, then $x$ was introduced by $\theta$ and thus in particular $x \in FV_a(\Theta(M))$ 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 rewrite 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 $l_i$ of $l$ at $p_i$. We have the following cases:

   - $x \in \Theta(l_i)$ is not possible since $\theta$ has the path condition for $M$ in $(l, r)$.

   - $x \notin \Theta(l_i)$:

     a) If $a \in \Theta(l_i)$, then $FV_a(\Theta(M)) = FV(\Theta(l))$ and thus the result obviously holds since $x \in FV(\Theta(M)) \subseteq FV(\Theta(l))$.

     b) If $a \notin \Theta(l_i)$, then we know that $x \in FV_a(\Theta(M)) \subseteq FV_a(\Theta(l))$ since $a \notin FV(l) \cup FV(r)$ by definition of well-formed metaterm.


LEMMA 2.11. (Preservation of localized free variables).

If \( s \rightarrow_R t \), then \( \forall a, \forall i, \mathcal{FV}_a^i(t) \subseteq \mathcal{FV}_a^i(s) \).

Proof. By induction on \( s \rightarrow_R t \) using Lemma 2.10.

COROLLARY 2.12. (Preservation of free variables). If \( s \rightarrow_R t \), then \( \mathcal{FV}(t) \subseteq \mathcal{FV}(s) \).

Proof. Take a choice variable \( a /\notin \mathcal{FV}(s) \cup \mathcal{FV}(t) \), use Lemma 2.11 and the facts that \( \mathcal{FV}_a^i(t) = \mathcal{FV}(t) \) and \( \mathcal{FV}_a^i(s) = \mathcal{FV}(s) \).

LEMMA 2.13. (Preservation of acceptability). If \( s \) is an acceptable term and \( s \rightarrow_R t \), then \( t \) is also acceptable.

Proof. By induction on \( s \rightarrow_R t \) using Lemma 2.11.

3. Examples

First of all let us show how reduction works on a simple example. For that, take the fibonacci function introduced just after Definition 1.1 and define a rewrite system to compute fibonacci numbers as follows:

\[
\begin{align*}
\text{fib}(M) & \rightarrow \text{app(FIB, M)} \\
\text{plus}(0, M) & \rightarrow M \\
\text{plus}(s(N), M) & \rightarrow s(\text{plus}(N, M)) \\
\text{app}(\lambda X . M, N) & \rightarrow M\{X \leftarrow N\}
\end{align*}
\]

where \( \text{FIB} = \text{def} \lambda a . \langle 0, s(0), s(s(x)) \rangle . a(0, s(0), \text{plus}(\text{fib}(x), \text{fib}(s(x)))) \).

As expected, we have the following reduction sequence:

\[
\begin{align*}
\text{app}(\text{FIB}, 2) & = \text{def} \\
\text{app}(\lambda a . \langle 0, 1, s(s(x)) \rangle . a(0, 1, \text{plus}(\text{fib}(x), \text{fib}(s(x)))) , 2) & \rightarrow \\
a(0, 1, \text{plus}(\text{fib}(x), \text{fib}(s(x))))\{a \leftarrow 3, x \leftarrow 0\} & = \\
\text{plus}(\text{fib}(0), \text{fib}(1)) & \rightarrow \\
\text{plus}(\text{app} (\text{FIB}, 0), \text{fib}(1)) & \rightarrow \\
\text{plus}(a(0, 1, \text{plus}(\text{fib}(x), \text{fib}(s(x))))\{a \leftarrow 1\}, \text{fib}(1)) & = \\
\text{plus}(0, \text{fib}(1)) & \rightarrow \\
\text{plus}(0, \text{app} (\text{FIB}, 1)) & \rightarrow \\
\text{plus}(0, a(0, 1, \text{fib}(x) + \text{fib}(s(x)))\{a \leftarrow 2\}) & = \\
\text{plus}(0, 1) & \rightarrow 1
\end{align*}
\]

We now consider the well known higher-order function \text{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 program:
#let rec map(f,l) = match l with
    [] -> []
  | h::t -> f(h)::map(f,t);;

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

\[ map(\mu X.F, L) \rightarrow \]
\[ a(\langle \text{nil}, \text{cons}\{F\{X\ by\ h\}, map(\mu X.F, t)\}\rangle\{a(\langle \text{nil}, \text{cons}(h, t)\rangle\ by\ L\})} \]

Let us see how this implementation of map works on a concrete example. Suppose that we represent natural numbers with constructors 0 and s and let us consider \( \text{pred} =_{\text{def}} \mu b(0, s(n)).b(0, n) \) in order to denote the predecessor function on natural numbers. Using the metasubstitution

\[ \text{theta} = \{ X \mapsto b(0, s(n)), F \mapsto b(0, n), L \mapsto \text{cons}(0, \text{cons}(s(0), \text{nil}))\} \]

to fire the previous rewrite rule, we can construct the following derivation:

\[ t_1 = \text{map}(\text{pred}, \text{cons}(0, \text{cons}(s(0), \text{nil}))) \rightarrow \]
\[ t_2 = \text{cons}(0, \text{map}(\text{pred}, \text{cons}(s(0), \text{nil}))) \]

Indeed, the term \( t_1 \) is a redex. In order to obtain \( t_2 \) we have to apply \( \text{theta} \) to the right-hand side of the previous rule. For that, we first instantiate \( \{ a(\langle \text{nil}, \text{cons}(h, t)\rangle\ by\ L)\} \) with \( \text{theta} \), then compute the pattern-matching operation

\[ \theta = \{ a(\langle \text{nil}, \text{cons}(h, t)\rangle\ by\ \text{cons}(0, \text{cons}(s(0), \text{nil}))\}\} \]

We obtain a substitution \( \theta = \{ a \mapsto 2, h \mapsto 0, t \mapsto \text{cons}(s(0), \text{nil})\} \).

Now, we have to instantiate \( a(\langle \text{nil}, \text{cons}(F\{X\ by\ h\}, map(\mu X.F, t)\}\rangle\{a(\langle \text{nil}, \text{cons}(h, t)\rangle\ by\ L\})\) with \( \text{theta} \), then proceed with the application of \( \theta \) to this last instantiation. The only delicate part is the one concerning the submetaterm \( F\{X\ by\ h\} \). We have \( \theta(F\{X\ by\ h\}) = b(0, n)\{b(0, s(n))\ by\ h\} = t' \) and \( \theta(t') = b(0, n)\{b(0, s(n))\ by\ 0\} = 0 \).

The reader may verify that this sequence of operations finally leads to the term \( t_2 \). Similarly, we can continue the reduction till the term \( \text{cons}(0, \text{cons}(s(0), \text{nil})) \).

Finally, let us consider another set of rewrite rules in order to illustrate the power of ERSP. The idea is to deal with complex patterns which can be seen either as the encoding of cut elimination rules in intuitionistic logic, or as evaluation rules of let expressions in functional languages with pattern-matching.
let \((M, \lambda X.let(M, \lambda Y.N))\) 
\(\rightarrow\) 
let \((M, \lambda X.\lambda Y.N)\) 
\(\rightarrow\) 
\(N\)

where \(let, pair, l, r, inl, inv\) are function symbols.

The interested reader is referred to (Cerrito and Kesner, 2004) for further details.

4. Towards a subclass of confluent ERSP

The following two sections are devoted to study of confluence for a certain class of ERSPs which are called the orthogonal \(l\)-constructor ERSPs, and a certain class of terms, which are called \(l\)-constructor deterministic terms. Intuitively, an orthogonal ERSP may only generate rewrite steps which may erase or copy other steps. It is well-known (Baader and Nipkow, 1998) that orthogonality is a sufficient condition to guarantee confluence in first and traditional higher-order rewrite systems. However, as we will see all along this section, orthogonality is not sufficient in ERSPs.

Now, let us introduce the notion of \(l\)-constructor ERSP. We partition the set of function symbols into two disjoint subsets, namely, the set of \textit{constructors}, which cannot be reduced, and the set of \textit{defined symbols}.

As an example, let us consider the following system which is not an \(l\)-constructor ERSP.

\[
\mathcal{R} : \begin{cases} 
  f \to g \\
  app(\mu f.h,M) \to h\{f \text{ by } M\}
\end{cases}
\]

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

Unfortunately, orthogonal \(l\)-constructor ERSP do not guarantee confluence as the rule \(app(\lambda X.M,N) \to M\{X \text{ by } N\}\) shows: the term \(t = app(\lambda a(x,y).a(0,1),3)\) has two non-joinable reducts 0 and 1. The reason is that \(t\) contains two “overlapping” subpatterns \(x\) and \(y\) inside the choice pattern \(a(x,y)\). Failure of confluence 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 now ready to give formal definitions.

**DEFINITION 4.1.** (L-constructor system). A system $R$ is said to be **l-constructor** iff

- The set $F$ of function symbols can be partitioned into two disjoint sets called **constructors** and **defined symbols**, such that:
  - Heads of left-hand sides of rules of $R$ are all defined symbols.
  - All the function symbols in metapatterns of $R$ are constructors.
- For every rule $(l, r) \in R$, both $l$ and $r$ are $p$-linear metaterms.

The following system is l-constructor:

$$R_1 = \begin{cases} 
\text{app}(\lambda X. M, N) \rightarrow_{\beta_{PM}} M\{X \text{ by } N}\} \\
\text{plus}(0, N) \rightarrow_{p_0} N \\
\text{plus}(s(M), N) \rightarrow_{p_s} s(\text{plus}(M, N))
\end{cases}$$

The systems $R_2 = \{\mu f(X). M \rightarrow M, f(0) \rightarrow 0\}$ and $R_3 = \{\mu f(X, X). 0 \rightarrow 0\}$ are not l-constructors.

**DEFINITION 4.2.** (L-constructor metapatterns and metaterms). Given an l-constructor system $R$, we say that a metapattern is l-constructor iff it is linear and all its function symbols are constructors of $R$. A **l-constructor metaterm** contains only l-constructor metapatterns.

As an example concerning our previous system $R_1$, we can observe that the metapattern $s(X)$ is l-constructor but $\text{plus}(X, Y)$ is not since the symbol $\text{plus}$ is not a constructor function symbol.

Even if Definition 4.2 depends on a given l-constructor system $R$ we will make an abuse of notation by just writing l-constructor metapattern/metaterm instead of $R$ l-constructor metapattern/metaterm.

**DEFINITION 4.3.** (L-constructor metasubstitutions). A metasubstitution $\sigma$ is said to be **l-constructor** w.r.t. a metaterm $t$ iff the first-order replacement $\sigma(t)$ is an l-constructor preterm.

A metasubstitution $\sigma$ is said to be **l-constructor** w.r.t. a rule $l \rightarrow r$ iff it is l-constructor for the metaterm $\mapsto (l, r)$ where $\mapsto$ is a fresh function symbol.
Consider again the previous system $R_1$. Then, the metasubstitution $\sigma = \{X \triangleright plus(0,0), M \triangleright 0, N \triangleright 0\}$ is not l-constructor for $\beta_{PM}$ but the metasubstitution $\theta = \{X \triangleright a(0,s(x)), M \triangleright a(0,x), N \triangleright plus(3,4)\}$ is l-constructor for it.

As explained in the introduction of this section, ERSPs need to be l-constructor in order to guarantee confluence. However, this is not sufficient since an l-constructor system which is instantiated by a non l-constructor replacement may break confluence anyway as the following non-joinable reductions in $R_1$ shows:

\[
\text{app}(\lambda plus(0,0), 3, \text{plus}(0,0)) \xrightarrow{\beta_{PM}} 3 \\
\text{app}(\lambda plus(0,0), 3, \text{plus}(0,0)) \xrightarrow{\alpha_{PM}} \text{app}(\lambda plus(0,0), 3, 0)
\]

where the replacement $\{X \triangleright plus(0,0), \ldots\}$ used to instantiate the $\beta_{PM}$ rule is not l-constructor.

In what follows, we give the necessary lemmas to guarantee that l-constructor systems instantiated by l-constructor replacements preserve l-constructor terms.

\textbf{Lemma 4.4.} Let $p$ be a pattern and $t$ be an l-constructor term. If $\{p \text{ by } t\}$ is defined, then for all $\sigma$ in $\{p \text{ by } t\}$ and for all $x$ in $\text{Dom}(\sigma)$, the term $\sigma(x)$ is l-constructor.

\textit{Proof.} The result is straightforward by induction on $p$.

\textbf{Lemma 4.5.} Let $t$ be an l-constructor preterm and let $\sigma$ be a substitution s.t. for all $x \in \text{Dom}(\sigma)$, the term $\sigma(x)$ is l-constructor. Then, if defined, the set $\sigma(t)$ only contains l-constructor terms.

\textit{Proof.} By induction on $t$. The only non trivial case is $t = t_1\{p \text{ by } t_2\}$. By $\alpha$-conversion we can suppose that $\text{Var}(p) \cap \text{Dom}(\sigma) = \emptyset$. Since $t_2$ is an l-constructor preterm then we know by i.h. that $\sigma(t_2)$ only contains l-constructor terms. Let $t_2'$ be a term in $\sigma(t_2)$. By Lemma 4.4, we know that for all $\theta \in \{p \text{ by } t_2'\}$, for all $x \in \text{Dom}(\theta)$, $\theta(x)$ is an l-constructor term. Since $t_1$ is an l-constructor preterm and the substitution $\theta \triangleright \sigma(t_1)$ verifies the hypothesis of the lemma, then by i.h. $(\theta \triangleright \sigma(t_1))$ only contains l-constructor terms which concludes the proof.

\textbf{Lemma 4.6.} Let $\sigma$ be an l-constructor metasubstitution w.r.t. the metaterm $t$. Then the set $\text{SIGMA}(t)$ only contains l-constructor terms.

\textit{Proof.} By definition $\text{SIGMA}(t) = \text{id}(\sigma(t))$. By hypothesis $\sigma(t)$ is an l-constructor preterm. By construction $\text{id}$ is a substitution verifying the hypothesis of Lemma 4.5. Then the same Lemma guarantees that $\text{id}(\sigma(t))$ only contains l-constructor terms.
DEFINITION 4.7. (L-constructor reduction relation). If the system $R$ is l-constructor, we say that $s$ (l-)constructor rewrites to $t$ (written $s \xrightarrow{c} R t$) iff there exists a rewrite rule $(l, r) \in R$, an l-constructor and admissible metasubstitution $\theta$ for $(l, r)$ and a context $C$ such that $s = C[\theta(l)]$ and $t \in C[\theta(r)]$.

Following again the system $R_1$, we have $s_0 = \text{app}(\lambda 0.3, 0) \xrightarrow{c} R_1 3$ and $s_1 = \text{app}(\lambda \text{plus}(0, 0).3, \text{plus}(0, 0)) \xrightarrow{a} R_1 3$ but we do not have $s_1 \xrightarrow{c} R_1 3$ since the metasubstitution $\{X \mapsto \text{plus}(0, 0)\}$ used to perform this reduction step is not l-constructor.

Using Definition 4.7, it is easy to show by induction on the definition of $\xrightarrow{c} R$ the following property.

LEMMA 4.8. (Preservation of l-constructor terms). If the system $R$ is l-constructor, $s$ is l-constructor and $s \xrightarrow{c} R t$, then $t$ is l-constructor.

Unfortunately again, l-constructor systems and replacements are not sufficient to guarantee confluence. Thus for example, consider again the l-constructor system $R_1$ which can be used to generate two non-joinable l-constructor reduction sequences:

$$\text{app}(\lambda x. a \langle x, y \rangle . a \langle 3, 4 \rangle, 0) \xrightarrow{c} 3$$
$$\text{app}(\lambda x. a \langle x, y \rangle . a \langle 3, 4 \rangle, 0) \xrightarrow{a} 4$$

The problem comes now from the fact that the pattern $a \langle x, y \rangle$ is non-deterministic, so that we will restrict our reduction relation to l-constructor deterministic terms for which the class of orthogonal l-constructor ERSP will be confluent. Let us start by the following notion.

DEFINITION 4.9. (Overlapping patterns). Two patterns $p$ and $q$ are said to be overlapping iff there exists a term $t$ s.t. both $\llbracket p \text{ by } t \rrbracket$ and $\llbracket q \text{ by } t \rrbracket$ are defined.

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

DEFINITION 4.10. (Deterministic patterns/preterms). 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 pairwise non overlapping, then $a\{p_1, \ldots, p_n\}$ is deterministic.

An acceptable preterm $t$ is said to be a deterministic preterm iff every pattern $p$ appearing in $t$ is deterministic.
Thus for example, $b(s(0), s(s(\_)))$ is deterministic but $b(s(0), s(\_))$ is not. We remark that if a term $t$ is deterministic then any subterm of $t$ is also deterministic.

**REMARK 4.11.** The definition of deterministic pattern implies that whenever $p$ is a deterministic pattern, then there exists at most one substitution $\theta$ belonging to $\{p \ by \ t\}$.

When $p$ is deterministic and $\{p \ by \ t\}$ is defined, we will identify $\{p \ by \ t\}$ with its single element.

**DEFINITION 4.12.** (Deterministic metasubstitutions). A metasubstitution $\theta$ is said to be deterministic for a metaterm $t$ iff

$-\theta$ is admissible for $t$,

-the first-order replacement $\theta(t)$ is a deterministic preterm,

Finally, $\theta$ is deterministic for a rule $(l, r)$ iff $\theta$ is a deterministic for the metaterm $\mapsto (l, r)$, where $\mapsto$ is a fresh function symbol.

**LEMMA 4.13.** If $p$ is a deterministic pattern, $t$ is a deterministic term and $\{p \ by \ t\}$ is defined, then for all $x \in \text{Dom}(\{p \ by \ t\})$ the term $\{p \ by \ t\}(x)$ is deterministic.

**Proof.** The proof is straightforward by induction on $p$.

**LEMMA 4.14.** Let $t$ be a deterministic preterm and $\sigma$ be a substitution s.t. $\forall x \in \text{Dom}(x)$ the term $\sigma(x)$ is deterministic. Then, if defined, $\sigma(t)$ is unique and deterministic.

**Proof.** The proof is done by induction on $t$. We only consider the non trivial case $t = t_1\{p \ by \ t_2\}$. By definition we know that $p$ is a deterministic pattern. Then, by i.h. $\sigma(t_2)$ is unique and deterministic. By Remark 4.11 and Lemma 4.13, we know that for all $x$ in $\text{Dom}(\{p \ by \ \sigma(t_2)\})$ the term $\{p \ by \ \sigma(t_2)\}(x)$ is deterministic. The result is then straightforward by i.h. on $t_1$ since $\sigma \uplus \{p \ by \ \sigma(t_2)\}$ verifies the hypothesis of the lemma.

**LEMMA 4.15.** Let $\sigma$ be a deterministic metasubstitution w.r.t. the metaterm $t$. Then $\Sigma(t)$, if defined, is unique and deterministic.

**Proof.** By definition $\Sigma(t) = \text{id}(\sigma(t))$. By hypothesis $\sigma(t)$ is a deterministic preterm. By construction $\text{id}$ is a substitution verifying the hypothesis of Lemma 4.14 so that $\text{id}(\sigma(t))$, if defined, is unique and deterministic.
DEFINITION 4.16. (Deterministic reduction relation). Given a system \( R \), we say that \( s \) \textbf{deterministically rewrites} to \( t \) (written \( s \rightarrow_{R} t \)) iff there exists a rewrite rule \((l, r) \in R\), a deterministic metasubstitution \( \theta \) for \((l, r)\) and a context \( C \) such that \( s = C[\Theta(l)] \) and \( s = C[\Theta(r)] \).

LEMMA 4.17. (Preservation of deterministic terms). Given a system \( R \), if \( s \) is deterministic and \( s \rightarrow_{R} t \), then \( t \) is deterministic.

\[ \text{Proof.} \quad \text{Since } s \text{ is in particular acceptable w.r.t. the used rule, then we know, by Lemma 2.13, that } t \text{ is also acceptable. We prove that each pattern } p \text{ appearing in } t \text{ is deterministic by induction on the definition of } s \rightarrow_{R} t. \]

\[- If \( s = \Theta(l) \) and \( t = \Theta(r) \) for some \( l \rightarrow r \in R \), then the result holds by Lemma 4.15.\]

\[- If \( s = \mu p.s' \) and \( t = \mu p.t' \) with \( s' \rightarrow_{R} t' \), let us consider a pattern \( p' \) appearing in \( t \). Only two cases are possible: \]

\[1. \quad p' \text{ is a subpattern of } p, \text{ then the result holds since } s \text{ is supposed to be deterministic.}\]

\[2. \quad p' \text{ appears in } t', \text{ then the result holds by i.h.}\]

\[- The other cases are obvious by i.h.\]

From now on we use the notation \( c \rightarrow_{R} c' \) to denote \( c \rightarrow_{R} \cap d \rightarrow_{R} c' \).

We give now the formal definition of orthogonality in our framework.

DEFINITION 4.18. (Left-linear systems). A rewrite rule \( l \rightarrow r \) is said to be \textbf{left-linear} iff \( l \) contains at most one occurrence of any term metavariable. A system is \textbf{left-linear} if all its rules are left-linear.

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

DEFINITION 4.19. (Redexes and overlapping redexes). Given a system \( R \) an \textbf{ERSP} and a relation \( \rightarrow_{\sim} \in \{ \rightarrow_{R}, \rightarrow_{d}, \rightarrow_{c}, \rightarrow_{c,d} \} \), a term \( t \) is said to be a \textbf{redex} for \( \rightarrow_{\sim} \) if \( t = \Theta(l) \) for some rule \((l, r) \in R\) and some \( \theta \) satisfying the conditions for \( \rightarrow_{\sim} \).

A rewrite system is said to be \textbf{non-overlapping} for \( \rightarrow_{\sim} \) iff
− Whenever a redex \( \text{THETA}(l_j) \) for \( \leadsto \) contains (not necessarily properly) another redex \( \text{RHO}(l_i) \) for \( \leadsto \) \((i \neq j)\), then \( \text{RHO}(l_i) \) must be contained in \( \text{THETA}(M) \) for some term metavariable \( M \) of \( l_j \).

− Likewise whenever a redex \( \text{THETA}(l) \) for \( \leadsto \) properly contains another redex \( \text{RHO}(l) \) for \( \leadsto \) of the same rule.

From now on, we will make an abuse of notation by simply saying that a term is a redex when the underlying reduction relation \( \leadsto \) is clear from the context.

**DEFINITION 4.20. (Orthogonal systems).** A rewrite system \( \mathcal{R} \) is said to be **orthogonal** \((w.r.t.\; \leadsto \in \{\alpha \rightarrow, \xi \rightarrow, d \rightarrow, c,d \rightarrow\})\) iff \( \mathcal{R} \) is left-linear and non-overlapping \((w.r.t.\; \leadsto)\).

From now on, we will make an abuse of notation by saying that a system is orthogonal when it is orthogonal \(w.r.t.\; c,d \rightarrow\).

As an example, the system \( \{f(\mu x.x) \rightarrow 0, \mu X.y \rightarrow 1\} \) is overlapping whatever \( \leadsto \) should be since the redex \( f(\mu y.y) \) contains the redex \( \mu y.y \). The system \( \{f(\mu X.M) \rightarrow 0, \lambda Z.N \rightarrow g(2)\} \) is orthogonal whatever \( \leadsto \) should be.

### 5. The confluence proof

This section shows that the relation \( c,d \rightarrow_{\mathcal{R}} \), which is only defined for l-constructor systems, is confluent for orthogonal ERSPs. The proof uses a technique due to Tait and Martin-Löf (Barendregt, 1984) and can be summarized in four steps:

− We define a “simultaneous” reduction relation denoted \( \gg_{c,d} \).

− We prove that \( \gg^{*}_{c,d} \) and \( c,d \rightarrow \) are the same relation.

− We show, using Takahashi terms (Takahashi, 1989), that \( \gg_{c,d} \) has the diamond property for orthogonal l-constructor ERSP.

− We conclude by the fact that the diamond property implies confluence.

In order to define the \( \gg_{c,d} \) reduction relation, we first need to extend relations on terms to relations on replacements and substitutions.
DEFINITION 5.1. (Relation between replacements/substitutions).

Given any relation \( \sim \) between terms and two replacements \( \theta \) and \( \rho \), we write \( \theta \sim \rho \) when \( \text{Dom}(\theta) = \text{Dom}(\rho) \), \( \theta \) and \( \rho \) coincide on the set of pattern metavariables, and for every term metavariable \( M \in \text{Dom}(\theta) \), \( \theta(M) \sim \rho(M) \).

Given two substitutions \( \theta \) and \( \rho \), we write \( \theta \sim \rho \) when \( \text{Dom}(\theta) = \text{Dom}(\rho) \), \( \theta \) and \( \rho \) coincide on the set of choice variables, and for every usual variable \( x \in \text{Dom}(\theta) \), \( \theta(x) \sim \rho(x) \).

LEMMA 5.2. (Preservation of the path condition). Let \( \theta \) be a metasubstitution and \( t \) be a metaterm such that \( \theta \) has the path condition for every term metavariable in \( t \). Let \( \rho \) be a metasubstitution such that \( \theta \overset{c,d}{\rightarrow} \rho \). Then, \( \rho \) has also the path condition for every term metavariable in \( t \).

Proof. Let us take a term metavariable \( M \) in \( t \). We know that \( \theta \) satisfies the path condition for \( M \) in \( t \), that is, if \( p_1 \ldots p_n \) are all the occurrences of \( M \) in \( t \) and \( l_1 \ldots l_n \) are their associated parameter paths:

\[
\forall 1 \leq i \leq n, \hat{x} \in \text{THETA}(l_i) \text{ or } \forall 1 \leq i \leq n, \hat{x} \notin \text{THETA}(l_i)
\]

Let us take \( \hat{x} \in \mathcal{FV}(\text{RHO}(M)) \). By hypothesis \( \theta(M) \overset{c,d}{\rightarrow} \rho(M) \), then by Corollary 2.12, \( \hat{x} \in \mathcal{FV}(\text{THETA}(M)) \) and hence the statement holds since \( \forall 1 \leq i \leq n, \text{THETA}(l_i) = \text{RHO}(l_i) \).

We can now define the simultaneous relation as follows:

DEFINITION 5.3. (The relation \( \gg_{c,d} \) on terms). The relation \( \gg_{c,d} \) on terms is defined as the smallest set satisfying the following clauses:

\[ x \gg_{c,d} x. \]

\[ - \text{If } s_1 \gg_{c,d} s'_1, \ldots, s_m \gg_{c,d} s'_m, \text{ then :} \]

\[ f(s_1, \ldots, s_m) \gg_{c,d} f(s'_1, \ldots, s'_m) \]

\[ - \text{If } s \gg_{c,d} s', \text{ then } \mu p.s \gg_{c,d} \mu p.s' \]

\[ - \text{If } s_1 \gg_{c,d} s'_1, \ldots, s_m \gg_{c,d} s'_m, \text{ then :} \]

\[ a(s_1, \ldots, s_m) \gg_{c,d} a(s'_1, \ldots, s'_m) \]

\[ - \text{If } \theta \gg_{c,d} \rho, \text{ then } \text{THETA}(l) \gg_{c,d} \text{RHO}(r) \text{ for any rule } l \rightarrow r \text{ such that } \theta \text{ is } l \text{-constructor and deterministic for it.} \]
If \( s \gg_{c,d} t \) is obtained with the last case, then we say that the simultaneous step is *external*. Otherwise, it is *internal*.

**REMARK 5.4.**

- For any term \( t \) we have \( t \gg_{c,d} t \).
- If \( s \gg_{c,d} s' \) and if \( s \) is not a redex then the head symbols of \( s \) and \( s' \) are the same. Moreover, if \( s = \mathcal{G}(s_1, \ldots, s_n) \) (with \( \mathcal{G} \) a function symbol, a choice constructor or an abstractor) then \( s' = \mathcal{G}(s'_1, \ldots, s'_n) \) and \( \forall i, s_i \gg_{c,d} s'_i \).

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 \( \rightarrow_{c,d} \) and \( \gg_{c,d} \) are the same relation.

For that, we first show that composition of substitution and metasubstitution can be expressed as metasubstitution (Lemma 5.5), pattern-matching is stable by term reduction (Lemma 5.6), application of substitution is stable by substitution reduction (Lemma 5.7), term reduction (Lemma 5.8) and metasubstitution reduction (Lemma 5.9).

**LEMMA 5.5.** Let \( \theta \) be a metasubstitution, \( t \) be a metaterm s.t. \( \Theta(t) \) is defined and \( \theta \) be a substitution s.t. \( \text{Dom}(\theta) \cap \text{FV}(t) = \emptyset \). If \( \rho \) is a metasubstitution verifying

- \( \text{Dom}(\rho) = \text{Dom}(\theta) \),
- \( \rho \) is equal to \( \theta \) on the pattern metavariables of \( t \),
- \( \rho(M) = \theta(\theta(M)) \) holds for any term metavariable \( M \) in \( \text{Dom}(\theta) \),

then, \( \rho(t) \) is defined and \( \rho(t) = \theta(\Theta(t)) \).

**Proof.** We first remark that \( \theta(\theta(M)) \) is a term (and not a set of terms) since \( \theta(M) \) has no pattern matching constructor by definition. Therefore, \( \rho(M) \) is well-defined. The proof proceeds by induction on \( t \).

- If \( t = M \) the result is straightforward by definition of \( \rho \).
- If \( t = x \) the result is straightforward since \( x \notin \text{Dom}(\theta) \) by hypothesis.
- If \( t = f(t_1, \ldots, t_n) \), \( t = a(t_1, \ldots, t_n) \) or \( t = \mu p.u \) the result is straightforward by i.h. (using the fact that \( a \notin \text{Dom}(\theta) \) when \( t = a(t_1, \ldots, t_n) \)).
− If \( t = u \{ p \ by \ v \} \), then, by i.h., we know that \( \text{RHO}(v) = \theta(\text{theta}(v)) \) and \( \text{RHO}(u) = \theta(\text{theta}(u)) \). By definition of rho, rho(p) = theta(p) (and we can suppose that rho(p) does not share variables with \( \theta \) by \( \alpha \)-conversion). Hence, it is straightforward by induction on \( p \) that \( \{ \text{RHO}(p) \ by \ \text{RHO}(v) \} \) is defined. Moreover if

\[
\{ \text{THETA}(p) \ by \ \text{THETA}(v) \} = \{ x_1 \triangleright t_1, \ldots, x_n \triangleright t_n \}
\]

then

\[
\{ \text{RHO}(p) \ by \ \text{RHO}(v) \} = \{ x_1 \triangleright \theta(t_1), \ldots, x_n \triangleright \theta(t_n) \}
\]

Thus, RHO(t) is defined and

\[
\text{RHO}(t) = \left( \{ x_1 \triangleright \theta(t_1), \ldots, x_n \triangleright \theta(t_n) \} \right) \left( \theta(\text{theta}(u)) \right) = \theta(\{ \text{THETA}(p) \ by \ \text{THETA}(v) \} (\theta(\text{theta}(u)))
\]

Hence the result holds.

**Lemma 5.6.** Let \( \mathcal{R} \) be an l-constructor ERSP, \( p \) be a deterministic and l-constructor pattern and \( t \) and \( t' \) be two terms such that \( c.d \stackrel{c.d}{\longrightarrow}_{\mathcal{R}} t' \). If \( \{ \ P \ by \ T \} \) is (uniquely) defined, then \( \{ \ P \ by \ T' \} \) is also defined (and is a singleton) and \( \{ \ P \ by \ T \} \stackrel{c.d}{\longrightarrow}_{\mathcal{R}} \{ \ P \ by \ \theta(T') \} \).

**Proof.** The proof proceeds by induction on \( p \).

− If \( p = x \) or \( p = \_ \) the result is obvious.

− If \( p = f(p_1, \ldots, p_n) \) then by definition of \( \{ \ P \ by \ T \} \) we must have \( t = f(t_1, \ldots, t_n) \) with \( \{ p_i \ by \ t_i \} \) defined for all \( i \). Since \( p \) is an l-constructor pattern then we know that \( f \) is a constructor symbol and since \( \mathcal{R} \) is an l-constructor system then we know that no rule in \( \mathcal{R} \) is headed by a constructor symbol, so that none of the considered reductions of \( t \) can take place at the root of \( t \). Thus, \( t' = f(t'_1, \ldots, t'_n) \) with \( t_i \stackrel{c.d}{\longrightarrow}_{\mathcal{R}} t'_i \) for every \( i \). By i.h. we know that all the \( \{ p_i \ by \ t'_i \} \) are defined and that \( \{ p_i \ by \ t_i \} \stackrel{c.d}{\longrightarrow}_{\mathcal{R}} \{ p_i \ by \ t'_i \} \) for every \( i \). Let us first show that \( \{ p \ by \ T' \} \) is defined. We need to show that \( \bigcup_i \{ p_i \ by \ t'_i \} \) is defined but this immediately holds since \( p \) is a linear pattern. The fact that \( \{ p \ by \ t \} \stackrel{c.d}{\longrightarrow}_{\mathcal{R}} \{ p \ by \ T' \} \) is obvious by i.h.
− If \( p = @ (p_1, \ldots, p_n) \) then we proceed as in the previous case.

− If \( p = a \langle p_1, \ldots, p_n \rangle \), since \( p \) is deterministic there exists an unique \( i \) s.t. \( \{ p_i \ by \ t \} \) is defined and then \( \{ p \ by \ t \} = \{ a \mapsto i \} \ union \{ p_i \ by \ t \} \). By i.h., we know that \( \{ p_i \ by \ t' \} \) is uniquely defined and that \( \{ p_i \ by \ t \} \overset{c\,d}{\rightarrow} \{ p_i \ by \ t' \} \). We can then conclude since \( \{ p \ by \ t \} = \{ a \mapsto i \} \ union \{ p_i \ by \ t' \} \).

The previous lemma does not hold when the pattern \( p \) is not l-constructor. Indeed, let \( R = \{ g \mapsto h \} \) and let us take the non-linear pattern \( p_1 = f(g, g) \) and the "non-constructor" pattern \( p_2 = g \).

\[ \{ p_1 \ by \ f(g, g) \} \] is defined and \( f(g, g) \overset{c\,d}{\rightarrow} R f(g, h) \) but \( \{ p_1 \ by \ f(g, h) \} \) is not defined. Also, \( \{ p_2 \ by \ g \} \) is defined and \( g \overset{c\,d}{\rightarrow} R h \) but \( \{ p_2 \ by \ h \} \) is not defined.

**Lemma 5.7.** Let \( R \) be an l-constructor ERSP and \( \theta, \rho \) be two substitutions such that \( \theta \overset{c\,d}{\rightarrow} R \rho \). If \( t \) is a deterministic l-constructor preterm such that \( \theta(t) \) is defined and for all \( x \in \text{Dom}(\theta) \) the term \( \theta(x) \) is l-constructor deterministic, then the same happens for \( \rho \) and \( \theta(t) \overset{c\,d}{\rightarrow} R \rho(t) \).

**Proof.** We reason by induction on the preterm \( t \). The only interesting case is when \( t = u \{ p \ by \ v \} \), where \( p \) is a deterministic and l-constructor pattern. Thus, by i.h. \( \rho(v) \) is defined and \( \theta(v) \overset{c\,d}{\rightarrow} R \rho(v) \). Since \( \{ p \ by \ \theta(v) \} \) is defined by hypothesis, then \( \{ p \ by \ \rho(v) \} \) is uniquely defined and \( \{ p \ by \ \theta(v) \} \overset{c\,d}{\rightarrow} \{ p \ by \ \rho(v) \} \), by Lemma 5.6. Thus:

\[ \theta \ union \{ p \ by \ \theta(v) \} \overset{c\,d}{\rightarrow} R \ union \{ p \ by \ \rho(v) \} \]

We can then conclude by i.h. on \( u \).

**Lemma 5.8.** If \( t \) and \( u \) are two terms such that \( t \overset{c\,d}{\rightarrow} u \), then for any substitution \( \theta \) s.t. for all \( x \in \text{Dom}(\theta) \) the term \( \theta(x) \) is l-constructor deterministic and \( \theta(t) \overset{c\,d}{\rightarrow} \theta(u) \).

**Proof.** By induction on the definition of \( \overset{c\,d}{\rightarrow} \). The only interesting case is when \( t = \text{THETA}(l) \) and \( u = \text{THETA}(r) \), where \( \text{theta} \) is an l-constructor deterministic metasubstitution for \( \mapsto (l, r) \). By Definition 2.1 the metaterms \( l \) and \( r \) are well-formed so that they do not contain free usual/choice variables.
Let us define a new metasubstitution $\rho$ having the same domain as $\theta$, being equal to $\theta$ on the pattern metavariables and such that for any term metavariable $M$ in $\text{Dom}(\theta)$, $\rho(M) = \theta(\theta(M))$.

Let us first remark that $\rho(l)$ and $\rho(r)$ are both defined since $\rho$ and $\theta$ have the same domain and $\theta(l)$ and $\theta(r)$ are both defined. As a consequence, $\text{RHO}(l) = \rho(l)$ is defined. The result now holds by Lemma 5.5.

As a consequence, $\text{RHO}(l) = \rho(l)$ is defined and, by Lemma 5.5, $\text{RHO}(r) = \theta(\theta(r))$. We then obtain by definition

$$\text{RHO}(l) = \theta(\theta(l)) \overset{c,d}{\rightarrow} \theta(\theta(r)) = \text{RHO}(r)$$

**Lemma 5.9.** Let $R$ be an l-constructor ERSP. Let $\theta, \rho$ be two metasubstitutions such that $\theta \overset{c,d}{\rightarrow}_R \rho$. Suppose that $\theta$ is l-constructor deterministic for the metaterm $t$ and $\text{THETA}(t)$ is defined. Then the same happens for $\rho$ and $\text{THETA}(t) \overset{c,d}{\rightarrow}_R \text{RHO}(t)$.

**Proof.** By Lemma 5.2, $\rho$ satisfies the path condition for $t$. So that to show that $\rho$ is an l-constructor deterministic metasubstitution for $t$, we only need to show that $\text{RHO}(t)$ only contains acceptable l-constructor deterministic terms. This point is shown by Lemmas 2.13, 4.8 and 4.17 and the fact that $\text{THETA}(t) \overset{c,d}{\rightarrow}_R \text{RHO}(t)$ for which we reason by induction on the metaterm $t$.

- For $t = x$, then $\text{THETA}(x) = \text{id}(\theta(x)) = \text{id}(x) = x$ and $\text{RHO}(x) = \text{id}(\rho(x)) = \text{id}(x) = x$ so we are done.

- For $t = M$, then $\theta(M) \overset{c,d}{\rightarrow}_R \rho(M)$ by hypothesis.

- For $t = f(t_1, \ldots, t_n)$, $t = a(t_1, \ldots, t_n)$, or $t = \mu p.t'$ the result is obvious by the i.h.

- For $t = u(p \text{ by } v)$ we know by hypothesis that $\text{THETA}(t)$ is defined. Moreover, we also know that $\text{THETA}(p) = \text{RHO}(p)$. By i.h., $\rho$ is l-constructor and deterministic for $v$ and $\text{THETA}(v) \overset{c,d}{\rightarrow} \text{RHO}(v)$. Also by hypothesis $\theta$ is l-constructor and deterministic for $t$ so that $\text{THETA}(p)$ is an l-constructor and deterministic pattern. We can then apply Lemma 5.6 which gives that $\langle \text{RHO}(p) \text{ by } \text{RHO}(v) \rangle$ is uniquely defined and that

$$\theta = \langle \text{THETA}(p) \text{ by } \text{THETA}(v) \rangle \overset{c,d}{\rightarrow} \langle \text{RHO}(p) \text{ by } \text{RHO}(v) \rangle = \rho$$
Also by i.h. we know that $\text{THETA}(u) \mapsto \text{RHO}(u)$. Hence, by Lemmas 5.8 and 5.7 we obtain

$$\theta(\text{THETA}(u)) \mapsto \theta(\text{RHO}(u)) \mapsto \rho(\text{RHO}(u))$$

which concludes the proof.

Lemma 5.9 allows us to obtain the following fundamental property.

**Lemma 5.10.** Let $\mathcal{R}$ be an $l$-constructor ERSP. If $\theta \mapsto_{\mathcal{R}} \rho$, and $\theta$ is an $l$-constructor deterministic metasubstitution for $(l, r)$, then so is $\rho$ and $\text{THETA}(l) \mapsto_{\mathcal{R}} \text{RHO}(r)$.

**Proof.** By hypothesis $\theta$ is $l$-constructor deterministic for $\mapsto (l, r)$ and thus by Lemma 5.9 $\rho$ is $l$-constructor deterministic for $\mapsto (l, r)$.

We have now to show $\text{THETA}(l) \mapsto_{\mathcal{R}} \text{RHO}(r)$, and in order to do that we proceed by induction on $\mu = \Sigma_{i=1}^{\Sigma} |\text{THETA}(M_i) \mapsto_{\mathcal{R}} \text{RHO}(M_i)|$, where $\text{Dom}(\theta) \cap TV = \{M_1, \ldots, M_n\}$.

- If $\mu = 0$, then $\rho = \theta$ and the property is trivial by definition of the rewriting relation.

- If $\mu > 0$, then there is $i$ such that $\text{THETA}(M_i) \mapsto_{\mathcal{R}} \text{RHO}(M_i)$. We then define a new metasubstitution $\xi$ by $\xi(M_j) = \text{RHO}(M_j)$ if $j \neq i$ and $\xi(M_i) = s$. We then have $\text{THETA}(l) \mapsto_{\mathcal{R}} \xi(r)$ by i.h. Since $\xi \mapsto_{\mathcal{R}} \rho$ holds by definition, then we can conclude $\xi(r) \mapsto_{\mathcal{R}} \text{RHO}(r)$ by Lemma 5.9 and thus $\text{THETA}(l) \mapsto_{\mathcal{R}} \text{RHO}(r)$.

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

**Theorem 5.11.** $(\mapsto_{\mathcal{R}} \implies \gg_{c,d})$. Let $\mathcal{R}$ be a ERSP and let $s$ be a term. If $s \mapsto t$ then $s \gg_{c,d} t$.

**Proof.** Let $s \mapsto t$. We prove $s \gg_{c,d} t$ by induction on the definition of $\mapsto$. 

30
\[ s = \text{THETA}(l) \text { and } t = \text{THETA}(r). \text { Since } \gg_{c,d} \text { is reflexive, then we have } \theta \gg_{c,d} \theta \text { and thus } s = \text{THETA}(l) \gg_{c,d} \text{THETA}(r) \text { by Definition 5.3.} \]

- For all the other cases the statement holds by i.h.

**THEOREM 5.12.** \((\gg_{c,d} \text{ implies } \rightarrow_{c,d}).\) Let the system \(R\) be an \(l\)-constructor ERSP and let \(s\) be a term. If \(s \gg_{c,d} t\) then \(s \rightarrow_{c,d} t.\)

*Proof. By induction on \(s \gg_{c,d} t.\)

- If \(s\) is a variable then the result is obvious.

- If \(s \gg_{c,d} t\) comes from the last point of Definition 5.3, then \(s = \text{THETA}(l), \ t = \text{RHO}(r)\) and \(\theta \gg_{c,d} \rho,\) where \((l, r) \in R\) and \(\theta\) is an \(l\)-constructor deterministic metasubstitution for \((l, r).\) We can suppose, without loss of generality, that the domain of \(\theta\) and \(\rho\) is restricted to the set of all the metavariables appearing in \((l, r).\) Since \(\text{THETA}(M)\) is smaller than \(s,\) then for every term metavariable \(M\) of \(\text{Dom}(\theta),\) we have by i.h. that \(\text{THETA}(M) \rightarrow_{c,d} \text{RHO}(M).\) By Lemma 5.10 we can thus conclude \(s = \text{THETA}(l) \rightarrow_{c,d} t = \text{RHO}(r).\)

- In all the other cases the statement holds by i.h.

We remark that the previous theorem does not hold in general as suggested by the example immediately after Lemma 5.6.

**COROLLARY 5.13.** Let \(R\) be an \(l\)-constructor ERSP and let \(s\) be a term. Then \(s \gg_{c,d} s' \text { iff } s \rightarrow_{c,d} s'.\)

We are now going to prove the diamond property for the relation \(\gg_{c,d}.\) For that, we associate a term \(#(s)\) to every term \(s\) such that every time \(s \gg_{c,d} s',\) for some \(s',\) we thus deduce \(s' \gg_{c,d} #(s).\) Thus, given two different terms \(s'\) and \(s''\) such that \(s \gg_{c,d} s'\) and \(s \gg_{c,d} s'',\) we obviously obtain a unique term \(#(s)\) which allows us to close the diagram with \(s' \gg_{c,d} #(s)\) and \(s'' \gg_{c,d} #(s).\) The diamond property will immediately follow.
DEFINITION 5.14. (Takahashi terms #\(s\)). Given a ERSP \(R\) and a term \(s\) we define its associated Takahashi term \#\(s\) by induction as follows:

1. If \(s = x\), then \#\(x\) = \(x\).
2. 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 redex w.r.t. \(c,d \rightarrow\), then \#\(s\) = \(f(#(s_1), \ldots, #(s_m))\) (resp. \(\mu p.#(s')\) and \(#(s) = a(#(s_1), \ldots, #(s_m))\)).
3. If \(s\) is a redex w.r.t \(c,d \rightarrow\), i.e. \(s\) can be written as \(\text{THETA}(l)\) for a rewrite rule \(l \rightarrow r\) and for an l-constructor deterministic meta-substitution \(\theta\), then \#\(s\) = \#(\text{THETA}(r))\), where \#(\theta) verifies \#(\theta)(M) = \#(\theta(M)) and \(#(\text{THETA}(r))\) stands for the application of the replacement \#(\theta) to \(r\).

We can show the following properties about \(\gg_{c,d}\) which are similar to Lemmas 5.6, 5.7, 5.9 and 5.10.

LEMMA 5.15. Let \(R\) be an l-constructor ERSP, \(p\) a deterministic and l-constructor pattern and \(t\) a term such that \(t \gg_{c,d} t'\). If \(\{p\ \text{by} \ t\}\) is defined, then \(\{p\ \text{by} \ t'\}\) is also defined (and is a singleton) and \(\{p\ \text{by} \ t\} \gg_{c,d} \{p\ \text{by} \ t'\}\).

Proof. Similar to the proof of Lemma 5.6.

LEMMA 5.16. Let the system \(R\) be an l-constructor ERSP and \(\theta, \rho\) two substitutions such that \(\theta \gg_{c,d} \rho\). If \(t\) is a deterministic and l-constructor preterm such that \(\theta(t)\) is defined and \(\forall x \in \text{Dom}(\theta), \theta(x)\) is an l-constructor deterministic term then the same happens for \(\rho\) and \(\theta(t) \gg_{c,d} \rho(t)\).

Proof. Similar to the proof of Lemma 5.7 using Lemma 5.15.

LEMMA 5.17. Let \(R\) be an l-constructor ERSP. Let \(\theta, \rho\) be two metasubstitutions such that \(\theta \gg_{c,d} \rho\). Suppose that \(\theta\) is an l-constructor deterministic for the metaterm \(t\) and \(\text{THETA}(t)\) is defined. Then the same happens for \(\rho\) and \(\text{THETA}(t) \gg_{c,d} \rho(t)\).

Proof. Similar to Lemma 5.9 using Lemma 5.16.

LEMMA 5.18. Let \(R\) be an l-constructor ERSP. If \(\theta \gg_{c,d} \rho\), and \(\theta\) is an l-constructor deterministic metasubstitution for \(l, r\), then we have \(\text{THETA}(l) \gg_{c,d} \rho(r)\) and \(\rho\) is an l-constructor deterministic metasubstitution for \(l, r\).
Proof. Similar to the proof of Lemma 5.10 using Lemma 5.17.

LEMMA 5.19. Let \( \mathcal{R} \) be an orthogonal \( l \)-constructor \( \mathcal{ERSP} \).

- The term \( \#(t) \) is uniquely defined for every term \( t \).
- \( t \gg_{c,d} \#(t) \) for every term \( t \).

Proof. The proof is done by induction on \( t \). The only non trivial case is when \( t \) is a redex. In that case we have a (unique by orthogonality) rewrite rule \((l,r) \in \mathcal{R}\) and an \( l \)-constructor and deterministic metasubstitution \( \sigma \) for \((l,r)\) such that \( t = \text{SIGMA}(l) \). We can suppose, without loss of generality that \( \text{SIGMA} \) is restricted to the set of all the metavariables of \((l,r)\). Now, for every term metavariable \( M \) in \( \text{Dom}(\sigma) \) we know that \( \text{SIGMA}(M) = \sigma(M) \) is structurally smaller than \( \text{SIGMA}(l) \), so that by i.h. we have:

- \( \#(\sigma(M)) \) is uniquely defined
- \( \sigma(M) \gg_{c,d} \#(\sigma(M)) \)

As a consequence, \( \sigma \gg_{c,d} \#(\sigma) \) according to Definition 5.1 so that \( t = \text{SIGMA}(l) \gg_{c,d} \#(\text{SIGMA})(r) = \#(t) \) by Definitions 5.14 and 5.3.

Remark that \( \sigma \gg_{c,d} \#(\sigma) \) implies \( \sigma \rightarrow_{\mathcal{R}} \#(\sigma) \) by Theorem 5.12. Therefore, the \( \#(\sigma) \) is an \( l \)-constructor metasubstitution and is deterministic for \( \rightarrow_{\mathcal{R}} \) \((l,r)\), by Lemma 5.9, so in particular \( \#(t) = \#(\sigma)(r) \) is uniquely defined.

LEMMA 5.20. Let \( \mathcal{R} \) be an \( l \)-constructor and orthogonal system and let \( s \) be a term. If \( s \gg_{c,d} s' \), then \( s' \gg_{c,d} \#(s) \).

Proof. The proof is by induction on \( s \). If \( s \) is a variable \( x \), then \( s' = x \) and thus \( \#(s) = x \) so that the property is trivial. Otherwise, two cases are possible.

- If \( s \gg_{c,d} s' \) is internal, then the property holds by i.h.
- If \( s \gg_{c,d} s' \) is external, then \( s = \text{THETA}(l) \) and \( s' = \text{RHO}(r) \) for a metasubstitution \( \theta \) which is \( l \)-constructor deterministic for a rule \((l,r) \in \mathcal{R}\) and such that \( \theta \gg_{c,d} \rho \).

We then have to show that \( s' \gg_{c,d} \#(s) = \#(\text{THETA})(r) \). By Lemma 5.17 we know that \( \rho \) is \( l \)-constructor deterministic for \((l,r)\). Let us show that \( \rho \gg_{c,d} \#(\theta) \).
For that, we know that for every term metavariable \( M \) in \( l \), \( \text{THETA}(M) \) is strictly smaller than \( \text{THETA}(l) \) and thus, by i.h., \( \text{RHO}(M) \gg_{c,d} \#(\text{THETA}(M)) \). As a consequence \( \text{rho} \gg_{c,d} \#(\text{theta}) \) and we conclude by Lemma 5.17 that:

\[
  s' = \text{RHO}(r) \gg_{c,d} \#(\text{THETA})(r) = \#(s)
\]

**THEOREM 5.21.** Let \( R \) be an \( l \)-constructor orthogonal ERSP. The relation \( \gg_{c,d} \) is confluent.

*Proof.* By Lemma 5.20 the relation \( \gg_{c,d} \) has the diamond property (i.e. for all \( t, u, v \) such that \( t \gg_{c,d} u \) and \( t \gg_{c,d} v \) there exists \( w \) such that \( u \gg_{c,d} w \) and \( v \gg_{c,d} w \)) and thus it is confluent (Baader and Nipkow, 1998).

**THEOREM 5.22.** Let \( R \) be an \( l \)-constructor orthogonal ERSP. The relation \( \rightarrow_{c,d}^{R} \) is confluent.

*Proof.* By Theorem 5.21 and Corollary 5.13.

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 complex patterns with metavariables. This formalism can be seen as an extension of ERSSs (Khasidashvili, 1990) and SERSs (Bonelli et al., 2000) to the case of patterns, and an extension of (Kesner et al., 1996) to the case of non functional rewrite rules. Many simple notions in the mentioned previous works do not trivially extend to our case: on one hand the complexity of ERSPs 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 \( \lambda \)-calculus. We carefully extend all the expected notions of rewriting to our framework, namely, terms, metaterms, rewrite rules, substitutions, reduction, etc. The resulting formalism allows to model definition of pattern-matching functions/proofs.

The more technical part of this work is the identification of a class of ERSPs which can be proved to be confluent on an appropriate set of terms. Our main result for ERSP gives in particular a confluence
result for SERSs. Indeed, SERSs can be viewed as a subclass of ERSPs where the metapatterns are only generated by usual variables and metaterms are never generated by the case constructor. Remark that restricting the ERSP syntax to SERS implies that (1) every (meta)term and (meta)substitution is acceptable, (2) admissibility is just restricted to the path condition, (3) every system is l-constructor and every (meta)term and metasubstitution is l-constructor, (4) all the terms and (meta)substitutions verifying the path condition are deterministic. An important consequence of all this is that the reduction relation \( \rightarrow \) coincides with \( \stackrel{c,d}{\rightarrow} \). Thus, orthogonality turns out to be the usual notion of orthogonality for SERSs and Theorem 5.22 simply states the SERS reduction relation generated by metasubstitutions respecting the path condition is confluent on orthogonal systems.

As mentioned in the introduction, ERSPs and \( \rho \)-calculus (Cirstea and Kirchner, 1998) are closely related. However, we can mention at least two fundamental differences. One of them lies in the class of syntactic metapatterns which are considered: quantification over patterns in ERSPs, via the notion of pattern metavariable, allows to create new binding relations when the rewrite rules are instantiated by the metasubstitutions. This is completely forbidden in \( \rho \)-calculus, where, even in recent versions allowing free variables inside patterns, the binding mechanism is completely static. Another important difference between the two formalisms is the treatment of non determinism: while the \( \rho \)-calculus collects different solutions of matching equations by using a special constructor to denote non-determinism, thus making the reduction relation naturally confluent, ERSPs allow to discard some terms via the use of choice variables, thus modelling divergence in a natural way.

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 the pattern-matching and the substitution operators are integrated to the syntax. This would result in generalizations of calculi defined in (Cerrito and Kesner, 2004; Forest, 2002).

Typing is another feature which remains as further work. It is however interesting to remark that work on pattern calculi (Kesner et al., 1996) 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 how
to associate a Curry-Howard style interpretation to the entire ERSP syntax.

Last but not least, strong normalization of ERSPs has to be studied. Indeed, proof techniques to guarantee termination of higher-order formalisms are not straightforward (Blanqui et al., 2001; Jouannaud and Rubio, 1999; van Raamsdonk, 2001) and they do not extend immediately to our case.

References


36