Developments in the rewriting calculus

Clara Bertolissi

Developments in the Rewriting Calculus

Clara Bertolissi
LORIA & UHP, Campus scientifique, Nancy, France
Clara.Bertolissi@loria.fr

January 8, 2007

Abstract

The theory of developments, originally developed for the λ-calculus, has been successfully adapted to several other computational paradigms, like first- and higher-order term rewrite system. The main desirable results on developments are the fact that the complete development of a finite set of redexes always terminates (FD) and the fact that, for a given initial term, all complete developments of a fixed set of redexes end with the same term (FD!). Following the ideas in the λ-calculus, in this paper, we present a notion of development and the proofs of theorems FD and FD! for the rewriting calculus, a framework embedding λ-calculus and rewriting capabilities, by allowing abstraction not only on variables but also on patterns. As an additional contribution, a new proof of the confluence property for the rewriting calculus, is obtained as a consequence of the results on developments.

1 Introduction

λ-calculus and term rewriting are known to be useful computational models for describing and analysing the behaviour of functional and rewrite-based programming languages. Properties of the original program can be inferred by studying the properties of its abstract model. For example, the fact that, for a given program, all its terminating computations end on the same result can be deduced by proving the confluence property of the rewrite relation in the associated abstract system. Proving global confluence its in general an hard task, but since [22] it is known that the proof can be decomposed in a proof of local confluence and a proof of termination of the rewrite relation. As nicely described in [21], originally termination is defined as the *strongly normalising* property (SN), ensuring that no object (term) in the abstract system can be infinitely rewritten. Local confluence plus SN lead to the confluence property of the simply typed λ-calculus, but the method cannot be applied to non strongly normalising systems, as the untyped λ-calculus. Therefore, a deeper analysis of the behaviour of non strongly normalising relations is carried out for the λ-calculus in [13, 16], in particular with respect to duplication phenomena during rewriting. Rewrite steps (redexes) are labelled in such a way that their copies (residuals) can be followed during the reduction. Information on the rewrite relation thus obtained allow to decompose the confluence property in a local confluence property plus a termination property known as the *finite developments* property (FD). The confluence of the non-typed λ-calculus, already proved in [5], can now be explained by abstract techniques, since even if the system is not strongly normalising, it verifies nevertheless the finite developments property. The work on decomposition of global confluence done for the λ-calculus has been successively generalised to other systems. In [23] an abstract method for proving the FD property in the setting of first-order term rewriting is proposed. In [19] the FD property has been generalised to an important class of

higher-order rewrite systems, namely the orthogonal Combinatory Reduction Systems (CRS). More recently, the FD property has been characterised as a set of axioms using a nesting relation between redexes for describing higher-order instantiation mechanisms [21]. This method is very general and it is shown to apply to λ-calculus, first- and higher-order rewrite systems.

In this paper we are interested in a system called rewriting calculus. The rewriting calculus (ρ-calculus, for short) has been introduced in the late nineties as a natural generalization of algebraic term rewriting and of λ-calculus [8]. The rewrite rules, acting as elaborated abstractions, their application and the obtained structured results are first class objects of the calculus. One essential component of the ρ-calculus are the matching constraints that are generated by the generalization of the β-reduction called ρ-reduction. The ρ-calculus has been shown to be a very expressive framework e.g. to express imperative languages and object calculi [20, 9] and it has been equipped with powerful type systems [2]. Several extensions of the calculus have already been studied, like the explicit ρ-calculus [17] and the graph rewriting calculus [3]. Therefore, the ρ-calculus can be considered a powerful formalism for specifying and reasoning about computations in functional programming. The operational semantics of the ρ-calculus was initially designed for modelling the execution of rewrite rules and strategies of the language ELAN [11]. Indeed, an important feature of the calculus is its ability to model rewriting strategies, thanks to the treatment of rewrite rules and sets of results as first-class citizens. Natural questions arise about the termination and confluence of such strategies. We are interested here in a better understanding of the behaviour of the calculus by analysing its derivation space. In particular, we will focus on the derivations called developments.

The first important contribution of this paper is the definition of developments in the setting of the rewriting calculus and the proof of the finiteness of such developments. ρ-developments are defined with respect to an underlined version of the calculus, as it is traditionally done in the λ-calculus. In order to prove the FD property for the ρ-calculus, we have chosen here to follow the approach proposed in [25] for the λ-calculus, which provides a rather short and elegant proof technique, exploiting an inductive characterisation of strongly normalising terms and the abstract notion of functional rewrite system morphism. We think that the abstract (quite technical) method proposed in [21] may be applied also to the ρ-calculus. We are analysing this possibility in parallel, in particular for standardisations issues, as discussed in the conclusions. We introduce also the notion of complete ρ-development which is given in terms of normal form *w.r.t.* the reduction relation of the underlined version of the calculus. The second theorem proved in this paper, called (FD!) using the Barendregt notation [1], states that all complete developments end with the same term.

The FD and FD! theorems have important consequences in the issues of confluence, standardisation and strong normalisation of typed systems. We treat in this paper the confluence of the ρ-calculus rewrite relation. This property has already been proved for an ancient version of the calculus in [6]. A confluence proof is available also for the typed version of the calculus called *PPTS* [2]. Here we provide an alternative proof for the ρ-calculus exploiting the finite development method. Finally, the presented results are obtained for the basic ρ-calculus, but can be adapted to the version of the ρ-calculus with explicit delayed matching constraints [10]. This version of the ρ-calculus is closer to concrete implementations and can be seen as the first step towards the explicit ρ-calculus [17]. Concluding, we think that the work described in this paper represents a first important achievement in the analysis of evaluation strategies for functional and rule-based programming languages based on the ρ-calculus.

The paper is organized as follows. In the following section, we recall some notions of abstract rewriting, in particular the notions of functional rewrite system and associated morphisms and liftings. Section 3 presents the basic rewriting calculus that will be used in the following two section. Section 4 provides the needed concepts for defining the ρ-developments that are shown to be finite in Section 5.

The obtained result is then used in Section 6 for proving the confluence of the ρ-calculus. In Section 7 we show how the work of the previous sections can be adapted to the ρ-calculus with explicit delayed matching constraints. The proofs of this part are collected in Section 8. We conclude in Section 9 by presenting some perspectives of future work.

2 Functional rewrite systems

A computation, intended as a stepwise transformation of some object (term), can mathematically be formalised as a set of objects $\mathcal{T}$ and a binary relation $\rightarrow$ on this set, that is a pair $(\mathcal{T}, \rightarrow)$. This model of computation is called *Abstract Rewrite System* (ARS). All term rewrite systems have an underlying abstract rewrite system. However, ARS give information on the results of computations, but cannot distinguish different ways of obtaining the same result. Since in rewriting we may be interested in tracing a property of a term along a rewrite sequence, it is useful to distinguish between computations. A solution can be to decorate computations by means of indices. This leads to the definition of *functional rewrite systems*, first introduced in [26].

Definition 1 (Functional rewrite systems)**.** *A* functional rewrite system *(or indexed abstract rewrite system) is a triple $(\mathcal{T}, \Delta, \rightarrow)$ where $\mathcal{T}$ is a set of objects, Δ a set of indices and $\rightarrow\colon \Delta \rightarrow [\mathcal{T} \rightharpoonup \mathcal{T}]$ a mapping from indices to partial functions from $\mathcal{T}$ to $\mathcal{T}$.*

Functional rewrite systems are appropriate to formalise in an abstract way the basic notions of rewriting.

Definition 2 (Redex, rewrite step and rewrite sequence)**.**

- *Let $t \in \mathcal{T}$. A* redex occurrence *in t is an index $i \in \Delta$ such that $\rightarrow (i)(t)$ is defined.*
- *A* redex *is a pair (t, i) such that i is a redex occurrence in t.*
- *A* rewrite step *in a functional rewrite system is denoted by $i : t \mapsto t'$ or $t \mapsto^i t'$ and defined as a triple (t, i, t') such that $\rightarrow (i)(t) = t'$.*
- *A* rewrite sequence *of lenght $n \in \mathbb{N}$ is a triple (t_0, n, r) such that $r : [0 \dots n] \rightarrow \Delta$ is a mapping that defines a sequence $t_0, t_1, \dots t_n$ as $t_m =\!\!\rightarrow (r(m))(t_{m-1})$ for all $m \leq n$, $m \neq 0$. We denoted such rewrite sequence by $t_0 \mapsto^{r(1)} t_1 \mapsto^{r(2)} \dots$*

Given two functional rewrite systems, we may be interested in establishing a correspondence between not only their objects, but also their derivations. This can be done using the notion of *morphism*. A morphism describes a correspondence that consists in fact of two parts: the correspondence between terms and the correspondence between rewrite steps.

Definition 3 (Morphism)**.** *Let $(\mathcal{R}, \Delta, \rightarrow_r)$ and $(\mathcal{S}, \Delta', \rightarrow_s)$ two functional rewrite systems. A* morphism *of functional rewrite systems is a mapping $\Theta = (\theta, \vartheta)$, where $\theta : \mathcal{R} \rightarrow \mathcal{S}$ and $\vartheta : \Delta \rightarrow \Delta'$, such that the rewrite sequences of the two systems correspond to each other in the following way:*

$$\begin{array}{ccc} i : t & \xmapsto{\quad r \quad} & t' \\ {\scriptstyle\theta}\downarrow & & \downarrow{\scriptstyle\theta} \\ \vartheta(i) : \theta(t) & \xmapsto{\quad s \quad} & \theta(t') \end{array}$$

For every rewrite step $i : t \mapsto_r t'$ in $(\mathcal{R}, \Delta, \rightarrow_r)$, we have a corresponding rewrite step $\vartheta(i) : \theta(t) \mapsto_s \theta(t')$ in $(\mathcal{S}, \Delta', \rightarrow_s)$. This correspondence can be generalised to rewrite sequences of arbitrary lenght in the obvious way.

In what follows, we will use morphisms to formalise the relationship between a "decorated" functional rewrite system, in the sense that its terms and rewritings contains some labels (indices or underlyinings for example), and the functional rewrite system obtained by erasing such decoration. A decorated rewrite sequence is often called a *lifting* of the rewrite sequence in the original rewrite system, obtained by erasing all decorations.

Definition 4 (Lifting)**.** *Let $\Theta = (\theta, \vartheta)$ be a morphism of two functional rewrite systems $(\mathcal{R}, \Delta, \rightarrow_r)$ and $(\mathcal{S}, \Delta', \rightarrow_s)$. Let r be a rewrite sequence in $(\mathcal{R}, \Delta, \rightarrow_r)$ and $\vartheta(r)$ be its image trough the morphism Θ in $(\mathcal{S}, \Delta', \rightarrow_s)$. Then r is said to be a* lifting *of $\vartheta(r)$.*

The notion of lifting will be used in Section 4 for defining developments in the ρ-calculus.

3 The Rewriting calculus

We briefly present in what follows the syntax and the semantics of the basic ρ-calculus. For a more detailed presentation the reader can refer to [8].

3.1 Syntax

In this paper, the symbols $t, u, \ldots$ range over the set $\mathcal{T}$ of terms, the symbols $x, y, z \ldots$ range over the infinite set $\mathcal{X}$ of variables and the symbols $f, g, \ldots$ range over the infinite set $\mathcal{F}$ of constants. Finally, the symbols p, q range over the set of patterns $\mathcal{P} \subseteq \mathcal{T}$. All symbols can be indexed. Syntactic equality is denoted by $\equiv$. We consider the meta-symbols "λ_._" (abstraction operator), and "_ $\wr$ _" (structure operator), and the (hidden) application operator. The set of ρ-terms is then defined as follows:

$$\begin{array}{lcl} \mathcal{T} & ::= & \mathcal{X} \mid \mathcal{F} \mid \lambda\mathcal{P}.\mathcal{T} \mid \mathcal{T}\ \mathcal{T} \mid \mathcal{T} \wr \mathcal{T} \\ \mathcal{P} & ::= & \mathcal{X} \mid \mathcal{F} \mid \mathcal{F}\ \mathcal{P}\ \mathcal{P} \end{array}$$

A term of the form $\lambda p.t$ is an *abstraction* with pattern p and body t. The term $t_1 \wr t_2$ is a *structure* consisting of the two terms t_1 and t_2. The set of patterns $\mathcal{P}$ is a parameter of the calculus and in full generality it could be as large as the set of all terms $\mathcal{T}$. We call *algebraic* the patterns used in this version of the calculus. A *linear* pattern is a pattern where every variable occurs at most once. In the rest of the paper we will consider only linear patterns. This restriction can be motivated by the fact that non-linearity is in general difficult to treat and it is a well-known source of problems for confluence results, as it is shown in [27].

We assume that the application operator associates to the left, while the other operators associate to the right. The priority of the application is higher than that of "λ_._" which is, in turn, of higher priority than the "_ $\wr$ _".

A term may be viewed as a *finite labeled tree*. Since we will later need them, we make now precise our definitions of sub-term and occurrence.

Definition 5 (Positions)**.** *A* position *(also called* occurrence*) of a term (seen as a tree) is a sequence ω of naturals describing the path from the root of t to the root of the sub-term at that position. For any term t we denote by $\mathcal{P}os(t)$ a sequence of natural numbers corresponding to the set of all possible positions in t. We denote ϵ the empty sequence indicating the head position of t. $t(\omega)$ denotes the symbol at position ω in t. A sub-term of t at position $\omega \in \mathcal{P}os(t)$ is denoted $t_{|\omega}$ and defined by $\forall \omega.\omega' \in \mathcal{P}os(t), \omega' \in \mathcal{P}os(t_{|\omega}), t_{|\omega}(\omega') = t(\omega.\omega')$. We use the notation $t_{\lceil u \rceil_\omega}$ to signify that t has a sub-term u at position ω.*

Similarly as in the λ-calculus, the $''\lambda\_.\_''$ operator is a binder of the calculus, *i.e.* in the term $\lambda p.t$ the free variables of p are bound in t. Formally:

Definition 6 (Free variables)**.**

$$\begin{array}{lcl} \mathcal{FV}(f) & = & \{\,\} \\ \mathcal{FV}(x) & = & \{x\} \\ \mathcal{FV}(t_1\, t_2) & = & \mathcal{FV}(t_1) \cup \mathcal{FV}(t_2) \\ \mathcal{FV}(t_1 \wr t_2) & = & \mathcal{FV}(t_1) \cup \mathcal{FV}(t_2) \\ \mathcal{FV}(\lambda p.t) & = & \mathcal{FV}(t) \setminus \mathcal{FV}(p) \end{array}$$

As in any calculus involving binders, we work modulo the α*-convention* of Church [4] and modulo the *hygiene-convention* of Barendregt [1].

Example 1 (ρ-terms)**.**

1. $(\lambda x.x\ x)\ (\lambda x.x\ x)$ *is the* ρ*-term corresponding to the* λ*-term* $\omega\ \omega$*;*
2. *The* ρ*-term* $(\lambda plus(x,0).x)\ plus(n,0)$ *encodes the application of the rewrite rule* $x+0 \to x$ *to the term* $n+0$*;*
3. *The* ρ*-term* $(\lambda f(a).a \ \wr\ \lambda f(a).b)$ *represents the rewrite system composed by the two rules* $f(a) \to a$ *and* $f(a) \to b$*.*

The classical notion of simultaneous substitution application used in higher-order calculi, like the λ-calculus, can be adapted to the ρ-calculus.

Definition 7 (Substitution)**.** *A substitution* σ *is a mapping from the set of variables to the set of terms. A finite substitution has the form* $\sigma = \{x_1/t_1 \ldots x_m/t_m\}$*, also denoted* $\sigma = \{\overline{x}/\overline{t}\}$*, where* $\mathcal{D}om(\sigma) = \{x_1, \ldots, x_m\}$*. The application of a substitution* σ *to a term* t*, denoted by* $\sigma(t)$ *or* $t\sigma$*, is defined as follows:*

$$\begin{array}{lcl} \sigma(f) & \triangleq & f \\ \sigma(x_i) & \triangleq & \left\{ \begin{array}{ll} t_i & \textit{if } x_i \in Dom(\sigma) \\ x_i & \textit{otherwise} \end{array} \right. \\ \sigma(\lambda p.t) & \triangleq & \lambda\sigma(p).\sigma(t) \\ \sigma(t_1\, t_2) & \triangleq & \sigma(t_1)\ \sigma(t_2) \\ \sigma(t_1 \wr t_2) & \triangleq & \sigma(t_1) \wr \sigma(t_2) \end{array}$$

We should point out that since we consider classes of terms modulo α-conversion, the appropriate representatives are always chosen in order to avoid potential variable captures.

We state here a lemma about the permutation of two substitutions that will be used in Section 6.

Lemma 1 (Substitution lemma)**.** *If* $x_i \neq y_j$ *and* $x_i \notin \mathcal{FV}(u_j)$ *for all* i, j *then*

$$t\{\overline{x}/\overline{v}\}\{\overline{y}/\overline{u}\} \equiv t\{\overline{y}/\overline{u}\}\{\overline{x}/\overline{v}\{\overline{y}/\overline{u}\}\}$$

Proof: By induction on the structure of t.

1. t is a variable. If $t = y_i \in \overline{y}$ then both sides equal t, since $x_i \notin \mathcal{FV}(u_j)$ for all i implies $t\{\overline{x}/\ldots\} \equiv t$. If $t = x_i \in \overline{x}$ then both sides equal $t\{\overline{y}/\overline{u}\}$, since $x_i \neq y_j$ for all i, j. If $t = z$ with $z \notin \overline{x}$ and $z \notin \overline{y}$, then both sides equal z.

2. $t = f$, then both sides equal f.
3. $t = \lambda p.u$ with $\mathcal{FV}(p) = \{z_1, \dots z_n\}$, $z_i \notin \overline{x}$ or $\overline{y}$ and z_i not free in $\overline{u}$, $\overline{v}$ by α-conversion, for all i. Then using the induction hypothesis
$$\begin{array}{rl} & (\lambda p.u)\{\overline{x}/\overline{v}\}\{\overline{y}/\overline{u}\} \\ = & \lambda p\{\overline{x}/\overline{v}\}\{\overline{y}/\overline{u}\}.u\{\overline{x}/\overline{v}\}\{\overline{y}/\overline{u}\} \\ \overset{ih}{=} & \lambda p\{\overline{y}/\overline{u}\}\{\overline{x}/\overline{v}\{\overline{y}/\overline{u}\}\}.u\{\overline{y}/\overline{u}\}\{\overline{x}/\overline{v}\{\overline{y}/\overline{u}\}\} \\ = & (\lambda p.u)\{\overline{y}/\overline{u}\}\{\overline{x}/\overline{v}\{\overline{y}/\overline{u}\}\} \end{array}$$
4. $t = t_1\ t_2$. We have:
$$\begin{array}{rl} & (t_1\ t_2)\{\overline{x}/\overline{v}\}\{\overline{y}/\overline{u}\} \\ = & t_1\{\overline{x}/\overline{v}\}\{\overline{y}/\overline{u}\}\ t_2\{\overline{x}/\overline{v}\}\{\overline{y}/\overline{u}\} \\ \overset{ih}{=} & t_1\{\overline{y}/\overline{u}\}\{\overline{x}/\overline{v}\{\overline{y}/\overline{u}\}\}\ t_2\{\overline{y}/\overline{u}\}\{\overline{x}/\overline{v}\{\overline{y}/\overline{u}\}\} \\ = & (t_1\ t_2)\{\overline{y}/\overline{u}\}\{\overline{x}/\overline{v}\{\overline{y}/\overline{u}\}\} \end{array}$$
5. $t = t_1 \wr t_2$. By induction hypothesis, similarly to 4..
6. $t = (\lambda p.u_1)\ u_2$ or $t = (\underline{\lambda} p.u_1)\ u_2$. By induction hypothesis, using 3. and 4..
7. $t = (t_1 \underline{\wr} t_2)\ t_3$. By induction hypothesis, using 4. and 5..

□

The evaluation mechanism of the calculus relies on the fundamental operation of *matching* that allows us to instantiate variables by their current values. We can use different matching theories for computing the matching substitutions like, for example, an empty theory, an equational theory or even more elaborated (higher-order matching) theories [9]. In this paper, we will restrict to syntactic matching problems, which are known to be decidable and unitary [18].

Definition 8 (Syntactic matching)**.** *A (syntactic) matching problem is a formula of the form $p \prec\!\!\!\prec t$, where p is a pattern and t is a term. A substitution σ is solution of the matching problem $p \prec\!\!\!\prec t$, denote by $Sol(p \prec\!\!\!\prec t)$, if $\sigma(p) \equiv t$.*

The small-step reduction semantics of the ρ-calculus is defined by the following reduction rules:

$$\begin{array}{lllll} (\rho) & (\lambda p.t_2)t_3 & \to_{\rho} & \sigma(t_2) & \textit{where } \sigma = Sol(p \prec\!\!\!\prec t_3) \\ (\delta) & (t_1 \wr t_2)\, t_3 & \to_{\delta} & t_1\, t_3 \wr t_2\, t_3 & \end{array}$$

The (ρ)-rule can be applied if (and only if) a substitution of the matching problem $p \prec\!\!\!\prec t_3$ exists. In this case, the result of the (ρ)-rule is the application of this substitution to the term t_2. If such a substitution does not exist, then the (ρ)-rule does not apply and the term is left as it is, representing a failure. Nevertheless, further reductions or instantiations are likely to modify t_3 so that the appropriate substitution can be found and the rule can be fired. The (δ)-rule right-distributes the application over the structures. This gives the possibility, for example, to apply in parallel two distinct pattern abstractions to a given term.

As usual, we introduce the classical notions of one-step, many-steps, and congruence with respect to the relation $\to_{\rho\delta}$ induced by the top-level rules of ρ-calculus. The one-step evaluation $\mapsto_{\rho\delta}$ is the contextual closure of $\to_{\rho\delta}$; if we want to specify the position ω at which the rewrite steps occurs, we write $\mapsto^{\omega}_{\rho\delta}$. The many-step evaluation $\mapsto\!\!\!\!\to_{\rho\delta}$ is defined as the reflexive and transitive closure of $\mapsto_{\rho\delta}$.

We characterise next the notions of non-reducibility and termination for a ρ-term t.

Definition 9 (Normalisation)**.**

- *A ρ-term t is in $\rho\delta$-normal form if there exists no term t' such that $t \mapsto\!\!\!\!\rightarrow_{\rho\delta} t'$.*
- *If every derivation starting from a ρ-term t reaches a normal form of t, then t is strongly normalising.*

Example 2 (Reductions)**.** *We consider the ρ-terms of Example 1 and and we show their respective reductions.*

1. $(\lambda x.x\ x)\ (\lambda x.x\ x) \mapsto_\rho (\lambda x.x\ x)\ (\lambda x.x\ x) \mapsto_\rho \ldots$ *is the infinite ρ-reduction corresponding to the reduction of the λ-term $\omega\ \omega$;*
2. *Since* $Sol(plus(x,0) \prec\!\!\!\prec plus(n,0)) = \{n/x\}$, *we have the reduction* $(\lambda plus(x,0).x)\ plus(n,0) \mapsto_\rho n$;
3. $(\lambda f(a).a \wr \lambda f(a).b)\ f(a) \mapsto_\delta$
$(\lambda f(a).a)\ f(a) \wr (\lambda f(a).b)\ f(a) \mapsto\!\!\!\!\rightarrow_\rho a \wr b$
is a ρ-reduction capturing the non-determinism of first-order term rewriting.

4 Developments in the ρ-calculus

Since developments are a special kind of reductions, which can be intuitively seen as a computation of a set of reducible expression in some term, we start by defining precisely the notion of *redex* (reducible expression, *w.r.t.* the evaluation rules (ρ) and (δ)) and of set of redex occurrences.

Definition 10 (Redex occurrence, redex)**.**

- *Let $t \in \mathcal{T}$. The pair $(\omega, \rho\delta)$ is a* $\rho\delta$-redex occurrence *in t if $t_{|\omega} = (t_0 \wr t_1)\ t_2$ or $t_{|\omega} = (\lambda p.t_1)\ t_2$ with $Sol(p \prec\!\!\!\prec t_2) \neq \emptyset$.*
- *The set $\Delta_{\rho\delta}$ consists of the pairs $(\omega, \rho\delta)$ where ω is a position. The set of all $\rho\delta$-redex occurrences in a term $t \in \mathcal{T}$ is denoted by $\Delta_{\rho\delta}(t)$.*
- *A* ρ-redex *is a pair $(t, (\omega, \rho\delta))$ such that $(\omega, \rho\delta)$ is a $\rho\delta$-redex occurrence in t.*

Notice that the ρ-calculus can be seen as a functional rewrite system $(\mathcal{T}, \Delta_{\rho\delta}, \rightarrow)$ with $\rightarrow (\omega, \rho\delta)(t) = t'$ if $t \mapsto^{\omega}_{\rho\delta} t'$ and undefined otherwise.

In order to be able to follow redexes along a rewrite sequence, it is useful to mark the initial redexes and observe their behaviour during the reduction. Traditionally, in the λ-calculus this is done using underlined (or indexed) terms. We follow here a similar approach and we introduce an underlined version of the ρ-calculus.

Definition 11 (Underlined ρ-calculus)**.** *The set $\underline{\mathcal{T}}$ of underlined ρ-term is inductively defined as:*

1. $x \in \underline{\mathcal{T}}$ *and* $f \in \underline{\mathcal{T}}$,
2. *if* $u_1, u_2 \in \underline{\mathcal{T}}$ *then* $(u_1 \wr u_2) \in \underline{\mathcal{T}}$,
3. *if* $u_1, u_2\ \underline{\mathcal{T}}$ *then* $(u_1\ u_2) \in \underline{\mathcal{T}}$,
4. *if* $p, t \in \underline{\mathcal{T}}$ *then* $\lambda p.t \in \underline{\mathcal{T}}$,
5. *if* $u_1, u_2, u_3 \in \underline{\mathcal{T}}$ *then* $(u_1 \underline{\wr} u_2)\ u_3 \in \underline{\mathcal{T}}$,

6. *if* $p, t, u \in \underline{\mathcal{T}}$ *and* $Sol(p \lessdot u) \neq \emptyset$ *then* $(\underline{\lambda} p.t)\ u \in \underline{\mathcal{T}}$,

The underlined version of the reduction rules is given by

$$\begin{array}{llll} (\underline{\rho}) & (\underline{\lambda} p.t_2)t_3 & \rightarrow_{\underline{\rho}} & \sigma(t_2) \quad \textit{where } \sigma = Sol(p \lessdot t_3) \\ (\underline{\delta}) & (t_1 \underline{\wr} t_2)\ t_3 & \rightarrow_{\underline{\delta}} & t_1\ t_3 \wr t_2\ t_3 \end{array}$$

The associated rewrite relations are denoted by $\mapsto_{\underline{\rho\delta}}$ *and* $\mapsto\!\!\!\!\rightarrow_{\underline{\rho\delta}}$.

Notice that only terms that are reducible expressions, that is structure applications or abstraction applications for which the matching is successful, are allowed to be underlined. We need to prove that the underlined ρ-calculus is well-defined, that is no illegal underlined terms appear during reductions. This follows from the next lemma, ensuring that the condition on the successful matching is preserved by reduction.

Lemma 2. *Let* p *be a* ρ*-pattern and* t, t' *be two* ρ*-terms such that* $t \mapsto\!\!\!\!\rightarrow_{\underline{\rho\delta}} t'$. *Then if* $\mathcal{S}ol(p \lessdot t) \neq \emptyset$ *we have* $\mathcal{S}ol(p \lessdot t') \neq \emptyset$.

Proof: A pattern p is by definition linear and algebraic. We prove first the statement for a one-step reduction $t_{\lceil r \rceil_\omega} \mapsto^{\omega}_{\underline{\rho\delta}} {t'}_{\lceil r' \rceil_\omega}$. The fact that $\mathcal{S}ol(p \lessdot t) \neq \emptyset$ implies that the redex occurrence $\omega \in \mathcal{P}os(t)$ corresponds to variables position in p, *i.e.* p is of the form $p_{\lceil x \rceil_\omega}$. Therefore, $\mathcal{S}ol(p_{\lceil x \rceil_\omega} \lessdot t_{\lceil r \rceil_\omega}) \neq \emptyset$ implies $\mathcal{S}ol(p_{\lceil x \rceil_\omega} \lessdot {t'}_{\lceil r' \rceil_\omega}) \neq \emptyset$. The reasoning can be easily iterated for a reduction sequence of length greater then one. □

Remark 1. *Definition 11 would not suit a* ρ*-calculus with non-linear patterns. Indeed, annoying phenomena, like non well-formed terms, would appear in the reduction of non-linear terms. For example, the one-step reduction of the non-linear underlined term* $t = (\underline{\lambda} f(x,x).x)\ \ f((\underline{\lambda} x.x)a, (\underline{\lambda} x.x)a)$ *would lead to a non well-formed term* $t' = (\underline{\lambda} f(x,x).x)\ f(a, (\underline{\lambda} x.x)a)$ *whose matching problem has no solution. Non-linearity matters will be further discussed in the conclusions.*

The definitions of position, free variables, substitution, matching and redex given for ρ-terms in Section 3, can be easily adapted to the underlined ρ-calculus. Moreover, like the ρ-calculus, the underlined ρ-calculus can be seen as a functional rewrite system $(\underline{\mathcal{T}}, \Delta_{\underline{\rho\delta}}, \rightarrow)$ with $\rightarrow(\omega, \underline{\rho\delta})(t) = t'$ if $t \mapsto^{\omega}_{\underline{\rho\delta}} t'$ and undefined otherwise.

In order to define developments, we need to formalise the correspondence between the ρ-calculus derivations and the underlined ρ-calculus derivations, respectively. For doing this, we use the functional rewrite system representation of the two calculi. We define a function Φ from the underlined ρ-calculus to the ρ-calculus that erases the underlinings and we show that Φ is a morphism of functional rewrite system.

Definition 12 (Erasing mapping)**.** *The mapping* $\Phi : (\phi, \varphi) : (\underline{\mathcal{T}}, \Delta_{\underline{\rho\delta}}, \rightarrow) \rightarrow (\mathcal{T}, \Delta_{\rho\delta}, \rightarrow)$ *is defined as*

1. *The mapping* $\phi : \underline{\mathcal{T}} \rightarrow \mathcal{T}$ *is defined by induction*

$$\begin{array}{lcllcl} \phi(x) & = & x & \phi(t_1 \wr t_2) & = & (\phi(t_1) \wr \phi(t_2)) \\ \phi(f) & = & f & \phi((t_1 \underline{\wr} t_2)\ t_3) & = & (\phi(t_1) \wr \phi(t_2))\ \phi(t_3) \\ \phi(t_1\ t_2) & = & \phi(t_1)\ \phi(t_2) & \phi(\lambda p.t_1) & = & \lambda\phi(p).\phi(t_1)) \\ & & & \phi((\underline{\lambda} p.t_1)\ t_2) & = & (\lambda\phi(p).\phi(t_1))\ \phi(t_2) \end{array}$$

2. *The mapping* $\varphi : \Delta_{\underline{\rho\delta}} \to \Delta_{\rho\delta}$ *is defined by* $\varphi((\omega, \underline{\rho})) = (\omega, \rho)$.

Lemma 3. *The mapping* $\Phi : (\phi, \varphi)$ *is a morphism of rewriting systems.*

Proof: We first observe that $\phi(t_{|\omega}) = \phi(t)_{|\omega}$. It follows that if we have a rewrite step $t \mapsto^{\omega}_{\underline{\rho\delta}} t'$, we have $\phi(t) \mapsto^{\omega}_{\rho\delta} \phi(t')$ in the ρ-calculus. □

Using the morphism Φ and the notion of *lifting* (Definition 4), we give the definitions of developments in the ρ-calculus.

Definition 13 (ρ-development)**.** *A* $\rho\delta$*-rewrite sequence* $r : t \mapsto\!\!\!\!\!\rightarrow_{\rho\delta} t'$ *is a* ρ*-development if there exists a* $\underline{\rho\delta}$*-rewrite sequence* r' *in* $(\underline{\mathcal{T}}, \Delta_{\underline{\rho\delta}}, \rightarrow)$ *that is a* Φ*-lifting of* r.

Definition 14 (Complete ρ-development)**.** *A* ρ*-development* $r : t_0 \mapsto\!\!\!\!\!\rightarrow_{\rho\delta} t_1$ *is a complete* ρ*-development if its lifting* r' *ends on a term* t'_1 *which is in normal form w.r.t. the relation* $\underline{\rho\delta}$.

5 Finiteness of ρ-developments

The goal of this section is to show that in the ρ-calculus all developments terminate. This is done following the method used for the λ-calculus in [25]. We make use of a set called SN that characterises strongly normalising ρ-terms and we show that developments corresponds to rewrite sequences in SN. This yields that all developments are finite.

Strongly normalising ρ-terms can be represented in a quite elegant way by a set SN, defined inductively. SN can be seen as the closure under expansion of the set of ρ-terms in normal form, with some restrictions in the way expansion is performed. In other words, a strongly normalising ρ-term is a term in normal form or can be obtained as the result of some expansion starting with a normal form.

Definition 15 (SN)**.** *The set* SN *is the set of all* ρ*-terms satisfying the following conditions:*

1. *if* x *is a variable and* $t_1, \ldots, t_n \in SN$ *then* $x\ t_1 \ldots t_n \in SN$,
2. *if* f *is a variable and* $t_1, \ldots, t_n \in SN$ *then* $f\ t_1 \ldots t_n \in SN$,
3. *if* $p, t \in SN$ *then* $\lambda p.t \in SN$,
4. *if* $u_1, u_2 \in SN$ *then* $(u_1 \wr u_2) \in SN$,
5. *if* $(u_1\ t_1 \wr u_2\ t_1)\ t_2 \ldots t_n \in SN$, *then* $(u_1 \wr u_2)\ t_1 \ldots t_n \in SN$,
6. *if* $p, u, t_i \in SN$ *for all* $i = 1, \ldots n$ *and* $Sol(p \lessdot u) = \emptyset$ *then* $(\lambda p.u)\ t_1 \ldots t_n \in SN$.
7. *if* $\sigma(u)\ \ t_1 \ldots t_n \in SN$ *with* $\sigma = \{\overline{x}/\overline{u}\} = Sol(p \lessdot t)$ *and* $u_i \in SN$, *for all* $u_i \in \overline{u}$, *then* $(\lambda p.u)\ t\ t_1 \ldots t_n \in SN$.

It is easy to observe that every subterm of a term in SN is in SN and that all ρ-terms in normal form are in SN, the base case being clauses 1. and 2. with $n = 0$. We prove next that SN is a correct characterisation of all strongly normalising ρ-terms.

Lemma 4. *A* ρ*-term* t *is strongly normalising iff* $t \in SN$.

Proof: Let $\texttt{MaxRed}(t)$ be the maximal length of the reduction from t to its normal form. The proof is done by induction on $(\texttt{MaxRed}(t), t)$ using the lexicographic product of the usual ordering on $\mathbb{N}$ and the subterm ordering.

We show first that t strongly normalising implies $t \in SN$. If $\texttt{MaxRed}(t) = 0$ then t is in normal form and it follows immediately that $t \in SN$.

If $\texttt{MaxRed}(t) > 0$, we have the following cases:

- $t = x\ t_1 \dots t_n$ or $t = f\ t_1 \dots t_n$. Then every reduct of t is of the form $x\ t'_1 \dots t'_n$ or $f\ t'_1 \dots t'_n$ where $t_i \mapsto\!\!\!\!\!\rightarrow_{\rho\delta} t'_i$ for all i. By induction hypothesis, $t_1 \dots t_n \in SN$ and hence by definition 15 we have $t \in SN$.
- $t = \lambda p.v$. Recall that a ρ-pattern p is in normal form by definition and thus $p \in SN$. Every reduct of t is of the form $\lambda p.v'$ with $v \mapsto\!\!\!\!\!\rightarrow_{\rho\delta} v'$. By induction hypothesis, v is in SN and hence by definition 15 we can conclude $t \in SN$.
- $t = u_1 \wr u_2$. Then every reduct of t is of the form $u'_1 \wr u'_2$ with $u_i \mapsto\!\!\!\!\!\rightarrow_{\rho\delta} u'_i$ for $i = 1, 2$. By induction hypothesis, $u_1, u_2 \in SN$ and hence by definition 15 we have $t \in SN$.
- $t = (\lambda p.u)\ t_1 \dots t_n$. We have two cases: a solution of the matching problem $p \lessdot t_1$ exists or not.

  If there exists no solution of the matching, by induction hypothesis p, u, t_i are SN, thus by definition 15 we conclude $t \in SN$. If there exists a substitution $\sigma = \{\overline{x}/\overline{u}\} = \mathit{Sol}(p \lessdot t_1)$, we have $t \mapsto_{\sigma} t' = \sigma(u)\ t_2 \dots t_n$. By induction hypothesis, $t' \in SN$ and $u_i \in SN$ for all $u_i \in \overline{u}$, thus by definition 15 we have $t \in SN$.
- $t = (u_1 \wr u_2)\ t_1 \dots t_n$. In this case, we can have the reduction $t \mapsto_{\delta} t' = (u_1\ t_1 \wr u_2\ t_1)\ t_2 \dots t_n$ and by induction hypothesis $t' \in SN$. Thus, by definition 15, we have $t \in SN$.

For the converse implication, we suppose $t \in SN$ and we show that t is strongly normalising.

- $t = x\ t_1 \dots t_n$ or $t = f\ t_1 \dots t_n$ with $t_1, \dots, t_n \in SN$, then by induction hypothesis $t_1, \dots, t_n$ are strongly normalising and hence we can conclude that t is strongly normalising.
- $t = \lambda p.u$ with $u \in SN$, then by induction hypothesis u is strongly normalising and thus also t is strongly normalising.
- $t = u_1 \wr u_2$ with $u_1, u_2 \in SN$, then by induction hypothesis u_1 and u_2 are strongly normalising and thus also t is strongly normalising.
- $t = (u_1 \wr u_2)\ t_1 \dots t_n$ with $(u_1\ t_1 \wr u_2\ t_1)\ t_2 \dots t_n \in SN$, then for an arbitrary rewrite sequence $t \mapsto\!\!\!\!\!\rightarrow_{\rho\delta} t' \mapsto\!\!\!\!\!\rightarrow_{\rho\delta} \dots$ we have two possibilities: either the head (δ) redex of t is contracted or not.

  If not, t reduce to a term $t' = (u'_1 \wr u'_2)\ t'_1 \dots t'_n$ such that $u_i \mapsto\!\!\!\!\!\rightarrow_{\rho\delta} u'_i$ and $t_i \mapsto\!\!\!\!\!\rightarrow_{\rho\delta} t'_i$, for all i. By induction hypothesis, all u_i and t_i are strongly normalising. It follows that all terms in the rewrite sequence are strongly normalising and thus the rewrite sequence is finite, *i.e.* t is strongly normalising.

  If the head (δ) redex is reduced, then $t' = (u'_1\ t'_1 \wr u'_2\ t'_1)\ t'_2 \dots t'_n$. By induction hypothesis, $(u_1\ t_1 \wr u_2\ t_1)\ t_2 \dots t_n$ is strongly normalising, thus its reduct t' is strongly normalising too. We conclude that the rewrite sequence is finite and t is strongly normalising.

- $t = (\lambda p.t_0)\ u\ t_1 \dots t_n$ with $\sigma(t_0)\ t_1 \dots t_n \in SN$ where $\sigma = \{\overline{x}/\overline{u}\} = Sol(p \lessdot u)$ and $u_i \in SN$ for all $u_i \in \overline{u}$, then for an arbitrary rewrite sequence $t \mapsto_{\rho\delta} t' \mapsto_{\rho\delta} \dots$ we have two possibilities: either the head (ρ) redex of t is contracted or not.
  If not, t reduce to a term $t' = (\lambda p.t'_0)\ u'\ t'_1 \dots t'_n$. By induction hypothesis p, u, t_i for all $i = 0 \dots n$ are strongly normalising. It follows that all terms in the rewrite sequence are strongly normalising and thus the rewrite sequence is finite, *i.e.* t is strongly normalising.
  If the head (ρ) redex is reduced, then $t' = \sigma(t'_0)\ \ t'_1 \dots t'_n$. By induction hypothesis, $\sigma(t_0)\ t_1 \dots t_n$ is strongly normalising, thus its reduct t' is also stronlgy normalising. We conclude that the rewrite sequence is finite and t is strongly normalising.

□

The correspondence between developments and derivations in SN is formalised by defining a morphism Θ that maps $\underline{\rho\delta}$-rewrite sequences in $\underline{\mathcal{T}}$ to $\rho\delta$-rewrite sequences in SN.

The morphism Θ erases the underlinings transforming all $\underline{\rho\delta}$-redexes in $\rho\delta$-redexes. However, this is not sufficient to ensure termination, since there can be $\rho\delta$-redexes that have no corresponding $\underline{\rho\delta}$-redexes and can lead to non terminating reductions. To avoid the problem, a distinguished constant, denoted Abs, is used to block this kind of $\rho\delta$-redexes. As a matter of fact, Abs is added in front of terms like $\lambda p.t$ or $t_1 \wr t_2$, in such a way that their application to a ρ-term will create no $\rho\delta$-redexes. We denote by $\mathcal{T}_{Abs}$ the target set of ρ-terms containing the constant Abs.

Definition 16 (Mapping Θ)**.** *The mapping $\Theta = (\theta, \vartheta) : (\underline{\mathcal{T}}, \Delta_{\underline{\rho\delta}}, \rightarrow) \rightarrow (\mathcal{T}_{Abs}, \Delta_{\rho\delta}, \rightarrow)$ is defined as*

1. *The mapping $\theta : \underline{\mathcal{T}} \rightarrow \mathcal{T}_{Abs}$ is defined by induction*

$$\begin{array}{lcllcl} \theta(x) & = & x & \theta(t_1 \wr t_2) & = & Abs(\theta(t_1) \wr \theta(t_2)) \\ \theta(f) & = & f & \theta((t_1 \underline{\wr} t_2)\ t_3) & = & (\theta(t_1) \wr \theta(t_2))\ \theta(t_3) \\ \theta(t_1\ t_2) & = & \theta(t_1)\ \theta(t_2) & \theta(\lambda p.t_1) & = & Abs(\lambda\theta(p).\theta(t_1)) \\ & & & \theta((\underline{\lambda} p.t_1)\ t_2) & = & (\lambda\theta(p).\theta(t_1))\ \theta(t_2) \end{array}$$

2. *The mapping $\vartheta : \Delta_{\underline{\rho\delta}} \rightarrow \Delta_{\rho\delta}$ is defined by $\vartheta((\omega, \underline{\rho\delta})) = (\omega, \rho\delta)$.*

Notation Let σ be the substitution $\{x_1/u_1, \dots, x_m/u_m\}$. Then $\theta(\sigma)$ denotes the substitution $\{x_1/\theta(u_1), \dots, x_m/\theta(u_m)\}$, also written $\{\overline{x}/\theta(\overline{u})\}$ for short.

The set of positions and the position of a subterm in a term $t \in \mathcal{T}_{Abs}$ is defined as for a ρ-term $t \in \mathcal{T}$ with the two additional clauses:

$$\mathcal{P}os(Abs(t)) = \mathcal{P}os(t) \quad \textit{and} \quad Abs(t)_{|\omega} = t_{|\omega}$$

This slightly modified definition is needed to make the mapping Θ a morphism of functional rewrite systems. We prove first two lemmata from which the proof of the morphism property for Θ will follow.

Lemma 5 (Context stability)**.** *Given the term $t \in \underline{\mathcal{T}}$, then for all $\omega \in \mathcal{P}os(t)$ we have*

$$\theta(t)_{|\omega} = \theta(t_{|\omega})$$

Proof: By induction on the position $\omega \in \mathcal{P}os(t)$.

If $\omega = \nu = \epsilon$, then the thesis follows easily by the definition of the mapping θ and the fact that $Abs(t)_{|\omega} = t_{|\omega}$.

If $\omega \neq \epsilon$, we show two interesting cases:

- $\omega = 2\omega'$ and $t = \lambda p.u$.

$$\begin{array}{rl} & \theta(t)_{|\omega} = \theta(\lambda p.u)_{|2\omega'} = Abs(\lambda\theta(p).\theta(u))_{|2\omega'} \\ = & \lambda\theta(p).\theta(u)_{|2\omega'} = \lambda\theta(p).\theta(u)_{|\omega'} \\ \stackrel{ih}{=} & \lambda\theta(p).\theta(u_{|\omega'}) = \theta(\lambda p.u_{|\omega'}) \\ = & \theta((\lambda p.u_{|2\omega'}) = \theta(t_{|\omega}) \end{array}$$

- $\omega = 2\omega'$ and $t = u_1 \wr u_2$.

$$\begin{array}{rl} & \theta(t)_{|\omega} = \theta(u_1 \wr u_2)_{|2\omega'} = Abs(\theta(u_1) \wr \theta(u_2))_{|2\omega'} \\ = & \theta(u_1) \wr \theta(u_2)_{|2\omega'} = \theta(u_1) \wr \theta(u_2)_{|\omega'} \\ \stackrel{ih}{=} & \theta(u_1) \wr \theta(u_{2|\omega'}) = \theta(u_1 \wr u_{2|\omega'}) \\ = & \theta((u_1 \wr u_{2|2\omega'}) = \theta(t_{|\omega}) \end{array}$$

□

Lemma 6 (Substitution stability)**.** *Given a term $t \in \underline{\mathcal{T}}$ and a substitution $\sigma = \{\overline{x}/\overline{u}\}$, then we have*

$$\theta(\sigma(t)) = \theta(\sigma)(\theta(t))$$

Proof: By structural induction on the term t.

- $t = x$ and $x \notin \mathcal{D}om(\sigma)$, then $\theta(\sigma(t)) = \theta(x) = x = \theta(t) = \theta(\sigma)(\theta(t))$
- $t = x$ and $x = x_i \in \mathcal{D}om(\sigma)$, then we have $\theta(\sigma(t)) = \theta(x\{\overline{x}/\overline{u}\}) = \theta(u_i) = x\{\overline{x}/\theta(\overline{u})\} = \theta(\sigma)(\theta(t))$
- $t = f$ then $\theta(\sigma(t)) = \theta(t) = \theta(\sigma)(\theta(t))$
- $t = \lambda p.u$ then

$$\begin{array}{rl} & \theta(\sigma(t)) = \theta(\lambda\sigma(p).\sigma(u)) = Abs(\lambda\theta(\sigma(p)).\theta(\sigma(u)) \\ \stackrel{ih}{=} & Abs(\lambda\theta(\sigma)(\theta(p)).\theta(\sigma)(\theta(u)) \\ = & \theta(\sigma)(Abs(\lambda\theta(p).\theta(u)) = \theta(\sigma)(\theta(\lambda p.u)) = \theta(\sigma)(\theta(t)) \end{array}$$

- $t = u_1 \wr u_2$ then

$$\begin{array}{rl} & \theta(\sigma(t)) = \theta(\sigma(u_1) \wr \sigma(u_2)) \\ = & Abs(\theta(\sigma(u_1)) \wr \theta(\sigma(u_2))) \\ \stackrel{ih}{=} & Abs(\theta(\sigma)\theta(u_1) \wr \theta(\sigma)\theta(u_2)) \\ = & \theta(\sigma)(Abs(\theta(u_1) \wr \theta(u_2))) \\ = & \theta(\sigma)(\theta(u_1 \wr u_2)) = \theta(\sigma)(\theta(t)) \end{array}$$

- $t = u_1\ u_2$ then

$$\begin{array}{rl} & \theta(\sigma(t)) = \theta(\sigma(u_1)\ \sigma(u_2)) = \theta(\sigma(u_1))\ \theta(\sigma(u_2)) \\ \stackrel{ih}{=} & \theta(\sigma)\theta(u_1)\ \theta(\sigma)\theta(u_2) = \theta(\sigma)(\theta(u_1)\ \theta(u_2)) \\ = & \theta(\sigma)(\theta(u_1\ u_2)) = \theta(\sigma)(\theta(t)) \end{array}$$

- $t = (u_1 \underline{\wr} u_2)\ u_3$ then

$$\begin{array}{rl} & \theta(\sigma(t)) = \theta((\sigma(u_1) \underline{\wr} \sigma(u_2))\ \sigma(u_3) \\ = & (\theta(\sigma(u_1)) \underline{\wr} \theta(\sigma(u_2)))\ (\theta(\sigma(u_3)) \\ \stackrel{ih}{=} & (\theta(\sigma)\theta(u_1) \underline{\wr} \theta(\sigma)\theta(u_2))\ (\theta(\sigma)\theta(u_3)) \\ = & \theta(\sigma)((\theta(u_1) \underline{\wr} \theta(u_2)))\ \theta(u_3) \\ = & \theta(\sigma)(\theta((u_1 \underline{\wr} u_2)\ u_3) = \theta(\sigma)(\theta(t)) \end{array}$$

- $t = (\underline{\lambda} p.u_1)\ u_2$ then

$$\begin{array}{rl} & \theta(\sigma(t)) = \theta((\underline{\lambda}\sigma(p).\sigma(u_1))\ \sigma(u_2)) \\ = & (\lambda\theta(\sigma(p)).\theta(\sigma(u_1))\ \theta(\sigma(u_2)) \\ \stackrel{ih}{=} & \lambda\theta(\sigma)(\theta(p)).\theta(\sigma)(\theta(u_1))\ \theta(\sigma)\theta(u_2) \\ = & \theta(\sigma)((\lambda\theta(p).\theta(u))\ \theta(u_2)) \\ = & \theta(\sigma)\theta((\underline{\lambda}p.u_1)\ u_2) = \theta(\sigma)(\theta(t)) \end{array}$$

□

Lemma 7. *The mapping $\Theta = (\theta, \vartheta)$ is a morphism of rewriting systems.*

Proof: By Lemma 5 and Lemma 6. □

We prove now that the morphism Θ maps an underlined ρ-term to a ρ-term in SN, *i.e.* a strongly normalising ρ-term.

Lemma 8. *Given any $t \in \underline{\mathcal{T}}$, then $\theta(t) \in SN$*

Proof: By structural induction on the term $t \in \underline{\mathcal{T}}$. We prove simultaneously that $\theta(t)$ is not of the form $\lambda p.u$ or $u_1 \wr u_2$.

- If $t = x$ or $t = f$, then $\theta(t) = x$ or $\theta(t) = f$ respectively and thus $\theta(t) \in SN$ follows immediately and $\theta(t)$ is not of the form $\lambda p.u$ or $u_1 \wr u_2$.
- If $t = \lambda p.u$ with $u, p \in \underline{\mathcal{T}}$, then $\theta(t) = Abs(\lambda\theta(p).\theta(t))$. By induction hypothesis $\theta(p)$ and $\theta(t)$ are in SN. By definition of SN it follows that $\theta(t) \in SN$. Moreover $\theta(t)$ is not of the form $\lambda p.u$ or $u_1 \wr u_2$.
- If $t = u_1 \wr u_2$ with $u_1, u_2 \in \underline{\mathcal{T}}$, then $\theta(t) = Abs(\theta(u_1) \wr \theta(u_2))$. By induction hypothesis $\theta(u_1)$ and $\theta(u_2)$ are in SN and thus $\theta(t) \in SN$. We have that $\theta(t)$ is not of the form $\lambda p.u$ or $u_1 \wr u_2$.
- If $t = u_1\ \ u_2$ with $u_1, u_2 \in \underline{\mathcal{T}}$, then $\theta(t) = \theta(u_1)\ \ \theta(u_2)$. By induction hypothesis $\theta(u_1)$ and $\theta(u_2)$ are in SN. Since $\theta(u_1)$ is neither $\lambda p.u$ nor $u_1 \wr u_2$, it follows easily that $\theta(t) \in SN$. Clearly $\theta(t)$ is not of the form $\lambda p.u$ or $u_1 \wr u_2$.
- If $t = (t_1 \wr t_2)\ \ t_3$ with $t_1, t_2, t_3 \in \underline{\mathcal{T}}$, then $\theta(t) = (\theta(t_1) \wr \theta(t_2))\ \ \theta(t_3)$. By induction hypothesis $\theta(t_1)$, $\theta(t_2)$, $\theta(t_3) \in SN$. It follows that the reduct $\theta(t') = (\theta(t_1)\ \ \theta(t_3) \wr \theta(t_2)\ \theta(t_3))$ is in SN and therefore we can conclude that also $\theta(t) \in SN$. Clearly $\theta(t)$ is not of the form $\lambda p.u$ or $u_1 \wr u_2$.
- If $t = (\underline{\lambda} p.t_2)\ \ t_1$ with $p, t_1, t_2 \in \underline{\mathcal{T}}$, then $\theta(t) = (\lambda\theta(p).\theta(t_2))\ \ \theta(t_1)$. By induction hypothesis $\theta(t_1)$, $\theta(t_2)$, $\theta(p) \in SN$. If a solution Let $\theta(\sigma) = \{\overline{x}/\theta(\overline{u})\}$ be the solution of the matching problem $\theta(p) \lessdot \theta(t_1)$. We have that $\theta(u_i) \in SN$ for all $u_i \in \overline{u}$ since $\theta(t_1) \in SN$ and moreover by induction hypothesis $\theta(u_i)$ are not of the form $\lambda p.u$ or $u_1 \wr u_2$. Therefore the reduct of t $\theta(t') = \theta(\sigma)(\theta(t_2))$ does not contain new redexes created by the application of the substitution to $\theta(t_2)$. Since $\theta(t_2) \in SN$, also $\theta(t') \in SN$ and thus $\theta(t) \in SN$. We have that $\theta(t)$ is not of the form $\lambda p.u$ or $u_1 \wr u_2$.

□

Since the morphism Θ ensures the correspondence between $\underline{\rho\delta}$-derivations and $\rho\delta$-derivations, finiteness of developments follows from the previous lemma.

Theorem 1 (FD)**.** *All ρ-developments are finite.*

Proof: By Lemma 7 and Lemma 8. □

Two interesting consequences can be deduced from the theorem FD. They will play an important role in the proof of confluence of the ρ-calculus by the method of finite developments, as described in the next section.

Corollary 1. *The $\underline{\rho\delta}$ relation is strongly normalising.*

Proof: Follows by Theorem 1 and Lemma 7. □

Corollary 2. *All ρ-developments of a ρ-term t can be extended to a complete ρ-development.*

Proof: Given any ρ-development $r : t \mapsto\!\!\!\!\twoheadrightarrow_{\rho\delta} t_1$, consider its lifting $r' : s \mapsto\!\!\!\!\twoheadrightarrow_{\underline{\rho\delta}} s_1$. Take a normalising extension of such derivation (which exists for Corollary 1) $r'' : s \mapsto\!\!\!\!\twoheadrightarrow_{\underline{\rho\delta}} s_1 \mapsto\!\!\!\!\twoheadrightarrow_{\underline{\rho\delta}} s'$, then the derivation in the ρ-calculus whose lifting is r'' is a complete ρ-development. □

6 Confluence of the ρ-calculus

The main result achieved so far is the finiteness of ρ-developments. In this section we will show an additional property of ρ-developments, namely that all complete ρ-developments terminate with the same term. These two properties will then be used to show the confluence of the ρ-calculus, following the so-called *finite development* method, as described for the λ-calculus [1].

The fact that all complete ρ-developments end with the same term corresponds in terms of the underlined ρ-calculus to the uniqueness of $\underline{\rho\delta}$-normal forms. In other words, the result on developments can be achieved by proving the confluence of the $\underline{\rho\delta}$ relation. This task is made easier by the normalisation result for the relation (see Corollary 1), that allow us to analyse simply its local confluence.

Lemma 9 (Local confluence of $\underline{\rho\delta}$)**.** *If $t \mapsto_{\underline{\rho\delta}} t_1$ and $t \mapsto_{\underline{\rho\delta}} t_2$, then there exists a ρ-term t_3 such that $t_1 \mapsto\!\!\!\!\twoheadrightarrow_{\underline{\rho\delta}} t_3$ and $t_2 \mapsto\!\!\!\!\twoheadrightarrow_{\underline{\rho\delta}} t_3$.*

Proof: We recall that any ρ-pattern p is linear and algebraic (thus in normal form). We analyse the different possible cases. The case in which the two redexes reduced in t are disjoint and the case in which the two redexes are the same redex are trivial. We analyse next the cases where the two redexes are contained one in the other. We use the symbol $\mathsf{Ctx}[\Box]$ for a context with exactly one hole $\Box$ and $\mathsf{Ctx}[t]$ for the ρ-term obtained by filling such a hole with t, defined in the obvious way.

- Two nested $\underline{\rho}$-redexes.
  1. $t = (\underline{\lambda} p.u)\ v$ and $u = \mathsf{Ctx}[(\underline{\lambda} p'.u')\ v']$. For readability, we consider in the proof the empty context. Suppose σ and σ' be the solutions of the two matching problems $p \lessdot v$ and $p' \lessdot v'$ respectively and let σ'' be the solution of the matching problem $p' \lessdot \sigma(v')$ (σ'' always exists by Lemma 2). We have $t \mapsto_{\underline{\rho}} t_1 = (\underline{\lambda} p.\sigma'(u'))\ v$ and $t \mapsto_{\underline{\rho}} t_2 = \sigma((\underline{\lambda} p'.u')\ v') = (\underline{\lambda} p'.\sigma(u'))\ \sigma(v')$. We obtain $t_1 \mapsto_{\underline{\rho\delta}} t_3 = \sigma(\sigma'(u'))$ and $t_2 \mapsto_{\underline{\rho\delta}} t'_3 = \sigma''(\sigma(u'))$. By the substitution lemma, we conclude $t_3 = t'_3$.

2. $t = (\underline{\lambda}p.u)\ v$ where $v = \mathsf{Ctx}[(\underline{\lambda}p'.u')\ s]$. For short, we denote the redex $(\underline{\lambda}p'.u')\ s$ by r and its contractum by r'. Let $\mathcal{FV}((u) = \{x_1, \ldots, x_n\}$ and $\omega_i \in \mathcal{P}os(u)$ be the position of x_i in u, if any, for $i = 1, \ldots m \leq n$. We have, on one hand, $t \mapsto_{\underline{\rho}} t_1 = (\underline{\lambda}p.u)\ v'$ with $v' = \mathsf{Ctx}[r']$ and, on the other hand, $t \mapsto_{\underline{\rho}} t_2 = u\{\overline{x}/\overline{v}\}$, with $\{\overline{x}/\overline{v}\}$ solution of the matching problem $p \lessdot v$. Notice that the second reduction is possible only if the position of the redex r in the term v corresponds to a variable position in p, *i.e.* there exists i and $v_i \in \overline{v}$ such that $v_i = r$, say for example $i = 1$. Thus we have $t_2 = u\{\overline{x}/\overline{v}\} = u_{\lceil r \rceil_{\omega_1} \ldots \lceil v_m \rceil_{\omega_m}}$. For the term t_1 we have the further reduction $t_1 \mapsto_{\underline{\rho}} t_3 = u\{\overline{x}/\overline{v'}\}$ with $\{\overline{x}/\overline{v'}\}$ solution of the matching problem $p \lessdot v'$. Observe that for all $v'_i \in \overline{v'}$ and $v_i \in \overline{v}$ with $i \neq 1$ we have $v'_i = v_i$; for $i = 1$ we have instead $v'_1 = r'$ and $v_1 = r$. Hence $t_3 = u_{\lceil r' \rceil_{\omega_1} \ldots \lceil v_m \rceil_{\omega_m}}$ and it is easy to see that $t_2 \mapsto_{\underline{\rho}} t_3$.

- Two nested $\underline{\delta}$-redexes. This case is easy to verify.
- A $\underline{\rho}$-redex and a non disjoint $\underline{\delta}$-redex. The interesting cases are those where the external redex is the $\underline{\rho}$-redex.
  1. $t = (\underline{\lambda}p.u)\ v$ with $v = \mathsf{Ctx}[(v_1 \wr v_2)\ v_3]$. This case is similar to the case 2. of the previous point.
  2. $t = (\underline{\lambda}p.u)\ v$ with $u = \mathsf{Ctx}[(u_1 \wr u_2)\ u_3]$. For simplicity, we show the proof for an empty context, the general case being similar. Let σ be the solution of the matching problem $p \lessdot v$. We have $t \mapsto_{\underline{\rho}} t_1 = \sigma(u)$ with $\sigma = \mathit{Sol}(p \lessdot v)$ and $t \mapsto_{\underline{\delta}} t_2 = (\underline{\lambda}p.u')\ v$ with $u \mapsto_{\underline{\delta}} u' = u_1\ u_3 \wr u_2\ u_3$. Then $t_2 \mapsto_{\underline{\rho}} t_3 = \sigma(u')$. It is not difficult to show that also $t_1 \mapsto_{\underline{\delta}} t_3$ holds. We have $t_1 = \sigma(u) = \sigma((u_1 \wr u_2)\ u_3) = (\sigma(u_1) \wr \sigma(u_2))\ \sigma(u_3) \mapsto_{\underline{\delta}} \sigma(u_1)\ \sigma(u_3) \wr \sigma(u_2)\ \sigma(u_3) = \sigma(u_1\ u_3 \wr u_2\ u_3) = \sigma(u') = t_3$.

□

The confluence property for the $\underline{\rho\delta}$ relation follows immediately from the termination and the local confluence of the relation [22].

Lemma 10 (Confluence of $\underline{\rho\delta}$)**.** *If $t \mapsto\!\!\!\!\!\rightarrow_{\underline{\rho\delta}} t_1$ and $t \mapsto\!\!\!\!\!\rightarrow_{\underline{\rho\delta}} t_2$, then there exists a ρ-term t_3 such that $t_1 \mapsto\!\!\!\!\!\rightarrow_{\underline{\rho\delta}} t_3$ and $t_2 \mapsto\!\!\!\!\!\rightarrow_{\underline{\rho\delta}} t_3$.*

Proof: By Lemma 9 and Corollary 1. □

We can reformulate this result in terms of developments.

Theorem 2 (FD!)**.** *All complete developments of a ρ-term t end on the same term.*

Proof: By Lemma 7 and Lemma 10. □

In the final part of the section, the obtained results FD and FD! on ρ-developments will be used for proving the confluence of the ρ-calculus. This proof method has been first used by Church and Rosser in order to prove the confluence of the λI-calculus [5]. Works on the full λ-calculus are developed in [16, 1]. More recently, different approaches exploiting the results on developments for showing the confluence of a rewrite relation have been proposed. For example, van Oomstrom [24] defines a notion of *consistency* and a construction called orthogonal projection that is used to prove that all consistent rewrite systems are confluent. Melliès [21] presents a multi-derivation space in which developments of a set of redexes are performed simultaneously in one step. The confluence of the original rewrite system can then be obtained reasoning over this abstract structure.

Here we prefer not to introduce the notions and the mathematical background needed in these last approaches. We will instead refer to the work done for the full λ-calculus in the presentation of Barendregt [1]. The idea consists, first, in defining a new rewrite relation, denoted Cpl. Intuitively, a step of Cpl rewriting on a ρ-term t consists in a complete development of a set of redexes initially fixed in t. Hence Cpl is defined using the lifting of a ρ-reduction in the underlined ρ-calculus and the erasing morphism $\Phi : (\phi, \varphi)$ described in Section 4. Exploiting the results on the ρ-developments, Cpl can be proved to enjoy the diamond property. We conclude then by showing that Cpl-reductions are good transcriptions of $\rho\delta$-derivations, in the sense that all $\rho\delta$-derivations can be obtained by transitivity from (finite) Cpl-reductions or, in other words, that the two relations have the same transitive closure.

Definition 17 (Cpl relation)**.** *Given two ρ-terms t_1 and $t_2 \in \mathcal{T}$, consider a set $\Delta_{\rho\delta}$ of redexes in t_1. Let t'_i be the underlined ρ-term such that $\phi(t'_1) = t_1$ and let $\Delta_{\underline{\rho\delta}}$ be the set of redexes of t'_1 such that $\varphi(\Delta_{\underline{\rho\delta}}) = \Delta_{\rho\delta}$. Then $t_1 \mapsto\!\!\!\!\!\rightarrow_{Cpl} t_2$ if its lifting $t'_1 \mapsto\!\!\!\!\!\rightarrow_{\underline{\rho\delta}} t'_2$ ends on t'_2 in $\underline{\rho\delta}$-normal form.*

Notice that Theorem 1 ensures that for every possible choice of redexes in t_1 we have a corresponding Cpl reduction.

The next goal is to prove the diamond property of the Cpl relation, from which the confluence of the $\rho\delta$-relation can be easily deduced.

Lemma 11 (Diamond of Cpl)**.** *If $t \mapsto\!\!\!\!\!\rightarrow_{Cpl} t_1$ and $t \mapsto\!\!\!\!\!\rightarrow_{Cpl} t_2$, then there exists a ρ-term t_3 such that $t_1 \mapsto\!\!\!\!\!\rightarrow_{Cpl} t_3$ and $t_2 \mapsto\!\!\!\!\!\rightarrow_{Cpl} t_3$.*

Proof: Given a term $t \in \mathcal{T}$ and a set of redexes $\Delta_{\rho\delta} = \Delta^1_{\rho\delta} \cup \Delta^2_{\rho\delta}$ of t, let $\Delta^i_{\underline{\rho\delta}}$ be such that $\varphi(\Delta^i_{\underline{\rho\delta}}) = \Delta^i_{\rho\delta}$ for $i = 1, 2$. We take t'_1 and $t'_2 \in \underline{\mathcal{T}}$ in $\underline{\rho\delta}$-normal form *w.r.t.* the set of redexes $\Delta^1_{\underline{\rho\delta}}$ and $\Delta^2_{\underline{\rho\delta}}$ respectively. Then we have $t \mapsto\!\!\!\!\!\rightarrow_{Cpl} \phi(t'_1) = t_1$ and $t \mapsto\!\!\!\!\!\rightarrow_{Cpl} \phi(t'_2) = t_2$. By developing the remaining redexes in $\Delta_{\rho\delta}$ we obtain by Theorem 1 $t_1 \mapsto\!\!\!\!\!\rightarrow_{Cpl} t_3$ and $t_2 \mapsto\!\!\!\!\!\rightarrow_{Cpl} t'_3$. By Theorem 2 we can conclude that $t_3 = t'_3$. □

Theorem 3 (Confluence of the basic ρ-calculus)**.** *The relation $\rho\delta$ is confluent.*

Proof: Notice that if the relation Cpl satisfies the diamond property, so does its transitive closure. The confluence of the $\rho\delta$ relation then follows easily from Lemma 11 by observing that the $\rho\delta$ relation and the Cpl relation have the same transitive closure. This can be shown using the inclusions $\mapsto_{\rho\delta} \subseteq \mapsto\!\!\!\!\!\rightarrow_{Cpl} \subseteq \mapsto\!\!\!\!\!\rightarrow_{\rho\delta}$. The second inclusion follows from the definition of the Cpl relation. For the first inclusion, suppose that the $\rho\delta$ step occurs at the redex position $(\omega, \rho\delta)$, then it is sufficient to choose $\Delta_{\rho\delta} = (\omega, \rho\delta)$. □

7 ρ-calculus with delayed matching

The results obtained for the basic ρ-calculus can be generalised to the ρ-calculus with delayed matching constraints, defined in [10]. This calculus introduces the matching problems as part of the ρ-calculus syntax and represents a first step towards an explicit handling of the matching related computations. More precisely, the *delayed matching constraints* represent constrained terms which are eventually instantiated by the substitution obtained as solution of the corresponding matching problem (if such a solution exists). Matching failures can be treated in different ways [9], in the current version of the calculus, the delayed matching constraints whose corresponding matching problem has no solution are considered in normal form. The ρ-calculus with delayed matching constraints has been

considered when equipping the ρ-calculus with dependent types, moreover it is an effective calculus for reasoning about functional languages implementations. It has been used as basis for the definition of the explicit ρ-calculus, for the encoding of the ρ-calculus into Interaction Nets [14] and for the compilation of functional languages with pattern matching features [7].

Basically, a ternary symbol $[\_ \ll \_]\_$, representing a term constrained by a matching, is added to the syntax of the ρ-calculus and the set of evaluation rules is adapted accordingly. The (ρ) rule is separated into two steps, the first step (still called (ρ)) transforms the matching into an explicit constraint and the second step (called (σ)) solves the matching and can be applied only if a substitution of such matching exists. As for the basic ρ-calculus, we will consider only syntactic matching. The (δ) rule remains unchanged.

$$\begin{array}{lllll} (\rho) & (\lambda p.t_2)t_3 & \to_\rho & [p \ll t_3]t_2 & \\ (\sigma) & [p \ll t_3]t_2 & \to_\sigma & \sigma(t_2) & \textit{where } \sigma = \mathit{Sol}(p \lessdot t_3) \\ (\delta) & (t_1 \wr t_2)\, t_3 & \to_\delta & t_1\, t_3 \wr t_2\, t_3 & \end{array}$$

The associated rewrite relations are denoted by $\mapsto_{\rho\sigma\delta}$ and $\mapsto\!\!\!\!\to_{\rho\sigma\delta}$. The new definition of *free variables* keeps into account the new binder $[\_ \ll \_]\_$ of the calculus, thus the clause

$$\mathcal{FV}([p \ll t_2]t_3) \triangleq (FV(t_3) \setminus \mathcal{FV}(p)) \cup \mathcal{FV}(t_2)$$

has to be added to Definition 6.

Definition 7 about *substitution* is generalised considering

$$\sigma([p \ll t_2]t_3) \triangleq [\sigma(t_1) \ll \sigma(t_2)]\sigma(t_3)$$

Recall that we work modulo α-conversion, thus the application of σ does not generate variable captures. The *Substitution Lemma* 1 can be shown to be still valid (see Appendix).

The definition of redexes is similar to Definition 10 for the basic ρ-calculus, considering now the rewrite relation $\rho\sigma\delta$. The only point that needs to be modified is the notion of *redex occurrence* since a new kind of redex is created, due to the introduction of the (σ)-rule.

Definition 18 (Redex occurrence, redex)**.**

- *A $\rho\sigma\delta$-redex occurrence in a ρ-term t is a pair $(\omega, \rho\sigma\delta)$ such that $t_{|\omega} = (\lambda p.t_1)\, t_2$ or $t_{|\omega} = (t_0 \wr t_1)\, t_2$ or else $t_{|\omega} = [p \ll t_1]t_2$ with $\mathit{Sol}(p \lessdot t_1) \neq \emptyset$.*
- *The set $\Delta_{\rho\sigma\delta}$ consists of the pairs $(\omega, \rho\sigma\delta)$ where ω is a position. The set of all $\rho\sigma\delta$-redex occurrences in a ρ-term t is denoted by $\Delta_{\rho\sigma\delta}(t)$.*
- *A* ρ-redex *is a pair $(t, (\omega, \rho\sigma\delta))$ such that $(\omega, \rho\sigma\delta)$ is a $\rho\sigma\delta$-redex occurrence in t.*

We point out that, similarly to the ρ-calculi previously introduced, the ρ-calculus with delayed matching constraints can be seen as a functional rewrite system $(\mathcal{T}, \Delta_{\rho\sigma\delta}, \to)$ with $\to (\omega, \rho\sigma\delta)(t) = t'$ if $t \mapsto^{\omega}_{\rho\sigma\delta} t'$ and undefined otherwise.

The delayed matching constraint and the new (ρ) and (σ) rules do not introduce substantial changes in the theory of developments for the rewriting calculus.

The reasoning proceeds as for the basic ρ-calculus. First, an underlined version of the calculus is defined. For doing this, we just need to add to Definition 11 the terms of the form $[\mathcal{P} \ll \mathcal{T}]\mathcal{T}$ and $[\mathcal{P} \underline{\ll} \mathcal{T}]\mathcal{T}$ to the set $\underline{\mathcal{T}}$ of underlined terms. The underlined version of the associated evaluation rules becomes the following:

$$
\begin{array}{lllll}
(\underline{\rho}) & (\underline{\lambda} p.t_2)t_3 & \to_{\underline{\rho}} & [p \underline{\ll} t_3]t_2 & \\
(\underline{\sigma}) & [p \underline{\ll} t_3]t_2 & \to_{\underline{\sigma}} & \sigma(t_2) \quad \textit{where } \sigma = \mathit{Sol}(p \prec\!\!\!\prec t_3) \\
(\underline{\delta}) & (t_1 \underline{\wr} t_2)\, t_3 & \to_{\underline{\delta}} & t_1\ t_3 \wr t_2\ t_3
\end{array}
$$

The mapping $\Phi : (\phi, \varphi)$ that erases the underlinings (Definition 12) can be trivially adapted to the ρ-calculus with delayed matching and it is easy to verify that it remains a morphism between the functional rewrite system corresponding to the underlined version of the calculus and the functional rewrite system representing the ρ-calculus with delayed matching.

Definitions 13 and 14 of ρ-development and complete ρ-development are essentially the same, considering now the relation $\rho\sigma\delta$ instead of $\rho\delta$.

Proofs of Theorem (FD) and Theorem (FD!) need to be revisited. As we have seen in Section 4, the proof of finite developments makes use of a set characterising strongly normalising ρ-terms, called SN, which needs to be adapted to the ρ-calculus with delayed matching constraints. Terms containing a matching constraint are strongly normalising if they are in normal form, *i.e.* a substitution of the matching does not exist, or they are obtained as an expansion, using the σ-rule, of a strongly normalising ρ-term.

Definition 19 (SN)**.** *The set SN is the set of all ρ-terms satisfying the first five conditions of Definition 15 plus the following ones:*

6. *if $\sigma(t)\ t_1 \dots t_n \in SN$ with $\sigma = \{x_i/u_i\} = \mathit{Sol}(p \prec\!\!\!\prec u)$ and $u_i \in SN$, for $i = 1 \dots n$, then $[p \ll u]t\ t_1 \dots t_n \in SN$,*
7. *if $p, u, t_i \in SN$ for all i and $\mathit{Sol}(p \prec\!\!\!\prec u) = \emptyset$ then $[p \ll u]t_1 \dots t_n \in SN$*
8. *if $[p \ll u]t\ t_1 \dots t_n \in SN$ then $(\lambda p.t)\ u\ t_1 \dots t_n \in SN$.*

The proof of Lemma 4, saying that ρ-term is strongly normalising if and only if it belongs to SN, is modified according to the new definition of SN. The reader can find the proof of this and the others lemmata in Section 8.

The key point in the proof of finite developments is the definition of the mapping $\Theta : (\theta, \vartheta)$ which, on one hand, has to map every underlined term to a strongly normalising ρ-term and, on the other hand, has to ensure a correspondence between the derivations of the underlined ρ-calculus and the ρ-calculus, respectively. From this two properties of Θ, the finiteness of developments is easy to deduce, since a development is a particular rewrite sequence in the underlined ρ-calculus.

For the ρ-calculus with delayed matching constraints, the definition of the mapping Θ is modified adding a new distinguished constant symbol $\mathtt{Bk}$ in the set of terms.

Definition 20 (Mapping Θ)**.** *The mapping $\Theta = (\theta, \vartheta) : (\underline{\mathcal{T}}, \Delta_{\underline{\rho\sigma\delta}}, \to) \to (\mathcal{T}_{Abs+\mathtt{Bk}}, \Delta_{\rho\sigma\delta}, \to)$ is defined as*

1. *The mapping $\theta : \underline{\mathcal{T}} \to \mathcal{T}_{Abs+\mathtt{Bk}}$ is defined as in Definition 16 with the two additional clauses*

$$
\begin{array}{lcl}
\theta([p \underline{\ll} t_1]t_2) & = & [\theta(p) \ll \theta(t_1)]\theta(t_2) \\
\theta([p \ll t_1]t_2) & = & [\mathtt{Bk}(\theta(\mathtt{p})) \ll \theta(\mathtt{t_1})]\theta(\mathtt{t_2})
\end{array}
$$

2. *The mapping $\vartheta : \Delta_{\underline{\rho\sigma\delta}} \to \Delta_{\rho\sigma\delta}$ is defined as in Definition 16 (considering $\underline{\rho\sigma\delta}$ and $\rho\sigma\delta$ instead of $\underline{\rho\delta}$ and $\rho\delta$).*

The symbol $\mathtt{Bk}$, similarly to the symbol Abs already introduced for the basic ρ-calculus, is used to block the evaluation of a (σ) redex $[p \ll t]u$ possibly generated by the (successful) matching problem $p \prec\!\!\!\prec t$.

In order to make the mapping Θ a morphism, we modify the set of positions and the position of a subterm in a term $t \in \mathcal{T}_{Abs+\mathtt{Bk}}$. In addition to the clauses for the constant Abs, already introduced for the basic ρ-calculus, we define

$$\mathcal{P}os(\mathtt{Bk(t)}) = \mathcal{P}os(\mathtt{t}) \quad \textit{and} \quad \mathtt{Bk(t)}_{|\omega} = \mathtt{t}_{|\omega}$$

Lemmata 7 and 8 are still valid and can be found in the Appendix. Therefore, Theorem FD (and its corollaries) holds also for the ρ-calculus with delayed matching constraints.

The second important result proved for the basic ρ-calculus (Theorem FD!) states that all complete developments of a ρ-term t end on the same term. This result is then used to conclude the confluence of the ρ-calculus.

The same reasoning applies to the proof of confluence of the $\rho\sigma\delta$ relation. The main lemma of this part of the proof is Lemma 9, stating the local confluence of the underlined $\underline{\rho\sigma\delta}$ relation (see Appendix), from which the the FD! result for the ρ-calculus with matching constraints follows. A relation Cpl with the same transitive closure as $\rho\sigma\delta$ is defined (as in Definition 17) and proved to enjoy the diamond property (Lemma 11) using the (FD) and (FD!) results. We therefore can conclude that the relation $\rho\sigma\delta$ is confluent.

Theorem 4 (Confluence of the ρ-calculus)**.** *The relation $\rho\sigma\delta$ is confluent.*

8 Proofs for general ρ-calculus

Lemma 12 (Substitution lemma)**.** *If $x_i \neq y_j$ and $x_i \notin \mathcal{FV}(u_j)$ for all i, j then*

$$t\{\overline{x}/\overline{v}\}\{\overline{y}/\overline{u}\} \equiv t\{\overline{y}/\overline{u}\}\{\overline{x}/\overline{v}\{\overline{y}/\overline{u}\}\}$$

Proof: By induction on the structure of t. We have the following cases, in addition to the ones shown for the basic ρ-calculus.

8 $t = [p \ll t_1]t_2$. We have

$$\begin{array}{rl} & ([p \ll t_1]t_2)\{\overline{x}/\overline{v}\}\{\overline{y}/\overline{u}\} \\ = & [p\{\overline{x}/\overline{v}\}\{\overline{y}/\overline{u}\} \ll t_1\{\overline{x}/\overline{v}\}\{\overline{y}/\overline{u}\}]t_2\{\overline{x}/\overline{v}\}\{\overline{y}/\overline{u}\} \\ \stackrel{ih}{=} & [p\{\overline{y}/\overline{u}\}\{\overline{x}/\overline{v}\{\overline{y}/\overline{u}\}\} \ll t_1\{\overline{y}/\overline{u}\}\{\overline{x}/\overline{v}\{\overline{y}/\overline{u}\}\}] \\ & t_2\{\overline{y}/\overline{u}\}\{\overline{x}/\overline{v}\{\overline{y}/\overline{u}\}\} \\ = & ([p \ll t_1]t_2)\{\overline{y}/\overline{u}\}\{\overline{x}/\overline{v}\{\overline{y}/\overline{u}\}\} \end{array}$$

9 $t = [p \underline{\ll} t_1]t_2$. By induction hypothesis, similarly to 8..

□

Lemma 13. *A ρ-term t is strongly normalising iff $t \in SN$.*

Proof: We need to consider only the following cases, the others being as in the basic ρ-calculus.

Direct implication:

- $u = (\lambda p.u')\ \ t_1 \dots t_n$. Then we have $t \mapsto_p t' = [p \ll t_1]u'\ \ t_2 \dots t_n$. By induction hypothesis, $t' \in SN$. By definition 19 it follows that $t \in SN$.
- $u = [p \ll u_1]u_2$. If there exists no solution of the matching $p \lessdot u_1$: by induction hypothesis p, u_1, u_2 are SN, thus by definition 19 $t \in SN$. If there exists $\sigma = \{x_i/u'_i\} = \mathit{Sol}(p \lessdot u_1)$, we have $t \mapsto_\sigma t' = \sigma(u_2)\ t_1 \dots t_n$. By induction hypothesis, $t' \in SN$ and $u'_i \in SN$, thus by definition 19 we have $t \in SN$.

Converse implication:

- If $t = (\lambda p.t_0)\ u\ t_1 \dots t_n$ with $[p \ll u]t_0\ t_1 \dots t_n \in SN$, then for an arbitrary rewrite sequence $t \mapsto_{\rho\sigma\delta} t' \mapsto_{\rho\sigma\delta} \dots$ we have two possibilities: either the head (ρ) redex of t is contracted or not.

  If not, t reduce to a term $t' = (\lambda p.t'_0)\ u'\ t'_1 \dots t'_n$. By induction hypothesis p, u, t_i for all i are strongly normalising. It follows that all terms in the rewrite sequence are strongly normalising and thus the rewrite sequence is finite, *i.e.* t is strongly normalising.

  If the head (ρ) redex is reduced, then $t' = [p \ll u']t'_0\ t'_1 \dots t'_n$. By induction hypothesis, $[p \ll u]t_0\ t_1 \dots t_n$ is strongly normalising, thus its reduct t' is stronlgy normalising too. We conclude that the rewrite sequence is finite and t is strongly normalising.
- If $t = [p \ll u]t_0\ t_1 \dots t_n$ with $\sigma(t_0)\ t_1 \dots t_n \in SN$ where $\sigma = \{x_i/u_i\} = \mathit{Sol}(p \lessdot u)$ and $u_i \in SN$. Then for an arbitrary rewrite sequence $t \mapsto_{\rho\sigma\delta} t' \mapsto_{\rho\sigma\delta} \dots$ we have two possibilities: either the head (σ) redex of t is contracted or not.

  If not, t reduce to a term $t' = [p \ll u']t'_0\ t'_1 \dots t'_n$. By induction hypothesis, p, u, t_i for all i are strongly normalising.It follows that all terms in the rewrite sequence are strongly normalising and thus the rewrite sequence is finite, *i.e.* t is strongly normalising.

  If the head (σ) redex is reduced, then $t' = \sigma(t'_0)\ \ t'_1 \dots t'_n$. By induction hypothesis, $\sigma(t_0)\ t_1 \dots t_n$ is strongly normalising, thus its reduct t' is stronlgy normalising too. We conclude that the rewrite sequence is finite and t is strongly normalising.
- If $t = [p \ll u]t_0\ \ t_1 \dots t_n$ with $p, u, t_i \in SN$ for all i and $\mathit{Sol}(p \lessdot u) = \emptyset$ then $t \mapsto_{\rho\sigma\delta} t' = [p \ll u']t'_0\ t'_1 \dots t'_n$ and we have two possibilities: either the matching has still no solution, or $\mathit{Sol}(p \lessdot u') \neq \emptyset$. In both cases, we can reason as in the previous item.

□

Lemma 14 (Context stability)**.** *Given the term $t \in \underline{\mathcal{T}}$, then for all $\omega \in \mathcal{P}os(t)$ we have*

$$\theta(t)_{|\omega} = \theta(t_{|\omega})$$

Proof: By induction on the position $\omega \in \mathcal{P}os(t)$. A part form the cases already adresses in the classic ρ-calculus, in the ρ-calculus with delayed matching constraints the new interesting case is the following:

- $\omega = 1\omega'$ and $t = [p \ll t_1]t_2$.

$$\begin{array}{rl} & \theta(t)_{|\omega} = \theta([p \ll t_1]t_2)_{|1\omega'} \\ = & [\mathtt{Bk}(\theta(\mathtt{p})) \ll \theta(\mathtt{t_1})]\theta(\mathtt{t_2})_{|1\omega'} \\ = & [\theta(p) \ll \theta(t_1)]\theta(t_2)_{|1\omega'} = [\theta(p)_{|\omega'} \ll \theta(t_1)]\theta(t_2) \\ \overset{ih}{=} & [\theta(p_{|\omega'}) \ll \theta(t_1)]\theta(t_2) = \theta([p_{|\omega'} \ll t_1]t_2) \\ = & \theta([p \ll t_1]t_{2|1\omega'}) = \theta(t_{|\omega}) \end{array}$$

□

Lemma 15 (Substitution stability)**.** *Given a term $t \in \underline{\mathcal{T}}$ and a substitution $\sigma = \{\overline{x}/\overline{u}\}$, then we have*

$$\theta(\sigma(t)) = \theta(\sigma)(\theta(t))$$

Proof: By structural induction on the term t.

In the ρ-calculus with delayed matching constraints, we have the following additional cases:

- $t = [p \ll u_1]u_2$ then
$$\begin{array}{rl} & \theta(\sigma(t)) = \theta([\sigma(p) \ll \sigma(u_1)]\sigma(u_2)) \\ = & [\mathtt{Bk}(\theta(\sigma(\mathtt{p}))) \ll \theta(\sigma(\mathtt{u_1})]\theta(\sigma(\mathtt{u_2})) \\ \stackrel{ih}{=} & [\mathtt{Bk}(\theta(\sigma)(\theta(\mathtt{p}))) \ll \theta(\sigma)(\theta(\mathtt{u_1})]\theta(\sigma)\theta(\mathtt{u_2}) \\ = & \theta(\sigma)([\mathtt{Bk}(\theta(\mathtt{p})).\theta(\mathtt{u_1})]\theta(\mathtt{u_2})) \\ = & \theta(\sigma)\theta([p \ll u_1]u_2) = \theta(\sigma)(\theta(t)) \end{array}$$
- $t = [p \lessll u_1]u_2$ then
$$\begin{array}{rl} & \theta(\sigma(t)) = \theta([\sigma(p) \lessll \sigma(u_1)]\sigma(u_2)) \\ = & [\theta(\sigma(p)) \lessll \theta(\sigma(u_1)]\theta(\sigma(u_2)) \\ \stackrel{ih}{=} & [\theta(\sigma)(\theta(p)) \lessll \theta(\sigma)(\theta(u_1)]\theta(\sigma)\theta(u_2) \\ = & \theta(\sigma)([\theta(p).\theta(u_1)]\theta(u_2)) \\ = & \theta(\sigma)\theta([p \ll u_1]u_2) = \theta(\sigma)(\theta(t)) \end{array}$$

□

Lemma 16. *Given any $t \in \underline{\mathcal{T}}$, then $\theta(t) \in SN$*

Proof: By structural induction on the term $t \in \underline{\mathcal{T}}$. We prove simoultaneously that $\theta(t)$ is not of the form $\lambda p.u$ or $u_1 \wr u_2$.

We add the following cases to the proof for the basic ρ-calculus.

- $t = [p \ll t_1]t_2$ with $p, t_1, t_2 \in \underline{\mathcal{T}}$, then $\theta(t) = [\mathtt{Bk}(\theta(\mathtt{p})) \ll \theta(\mathtt{t_1})]\theta(\mathtt{t_2})$. By induction hypothesis $\theta(t_1)$, $\theta(t_2)$, $\theta(p) \in SN$ and by definition of SN also $\mathtt{Bk}(\theta(\mathtt{p})) \in \mathtt{SN}$. Since there exists no solution σ for the matching $\mathtt{Bk}(\theta(\mathtt{p})) \lessdot \theta(\mathtt{t_1})$, we have $\theta(t) \in SN$. Moreover $\theta(t)$ is not of the form $\lambda p.u$ or $u_1 \wr u_2$.
- $t = [p \lessll t_1]t_2$ with $p, t_1, t_2 \in \underline{\mathcal{T}}$, then $\theta(t) = [\mathtt{Bk}(\theta(\mathtt{p})) \ll \theta(\mathtt{t_1})]\theta(\mathtt{t_2})$. By induction hypothesis $\theta(t_1)$, $\theta(t_2)$, $\theta(p) \in SN$. Let $\theta(\sigma) = \{x_i/u_i^\theta\}$ be the solution of the matching problem $\theta(p) \lessdot \theta(t_1)$. We have that $\theta(u_i) \in SN$ for all i since $\theta(t_1) \in SN$ and moreover by induction hypothesis $\theta(u_i)$ are not of the form $\lambda p.u$ or $u_1 \wr u_2$. Therefore the reduct of t $\theta(t') = \theta(\sigma)(\theta(t_2))$ does not contain new redexes created by the application of the substitution to $\theta(t_2)$. Since $\theta(t_2) \in SN$, also $\theta(t') \in SN$ and thus $\theta(t) \in SN$. We have that $\theta(t)$ is not of the form $\lambda p.u$ or $u_1 \wr u_2$.
- $t = (\underline{\lambda} p.t_1)\ \ t_2$ with $p, t_1, t_2 \in \underline{\mathcal{T}}$, then $\theta(t) = (\lambda\theta(p).\theta(t_1))\ \ \theta(t_2)$. By induction hypothesis $\theta(t_1)$, $\theta(t_2)$, $\theta(p) \in SN$. We reason on the reduct $\theta(t') = [\theta(p) \ll \theta(t_1)]\theta(t_2)$ as in the previous item. We deduce that $\theta(t') \in SN$, hence $\theta(t) \in SN$. We have that $\theta(t)$ is not of the form $\lambda p.u$ or $u_1 \wr u_2$.

□

Lemma 17. *The relation $\underline{\rho\delta}$ is locally confluent.*

Proof: We have to consider the possible critical pairs. The critical pairs generated by the ($\underline{\rho}$)-rule are easy to verify. We analyse the critical pairs between the ($\underline{\sigma}$)-rule with itself and with the ($\underline{\delta}$)-rule.

- Two nested $\underline{\sigma}$-redexes. These critical pairs are similar to the ones between two ($\underline{\rho}$)-rules treated in the basic ρ-calculus.
- A $\underline{\sigma}$-redex and a non disjoint ($\underline{\delta}$)-redex. These critical pairs are similar to the ones between the ($\underline{\rho}$)-rule and the ($\underline{\delta}$)-rule of the basic ρ-calculus.

□

9 Conclusions

We have presented in this paper the theory of developments for the rewriting calculus. After giving an appropriate definition of ρ-development, we have proved that in the ρ-calculus all ways of reducing a set of redexes of a ρ-term are equivalent: all development terminates (FD) and they all end on the same final ρ-term (FD!). As a consequence of these results, we obtained a new simpler proof for the confluence property of the ρ-calculus rewrite relation.

The mentioned results are achieved here for the basic ρ-calculus considering syntactic matching and linear algebraic patterns. A natural question is whether these results hold for other versions of the ρ-calculus. We have shown here that this is indeed the case for the ρ-calculus with delayed matching constraints. It would be interesting to go a step further and study two calculi that extend this version of the ρ-calculus, that is the explicit rewriting calculus and the graph rewriting calculus. Nevertheless, we think that the consistent number of evaluation rules of these two calculi will make the proof of the result more elaborated. We are also interested in ρ-calculi with non syntactic matching theories, like, for example, the ρ-calculus with delayed matching constraints and the matching theory called *stuck* [12], which eliminates matching failures when not significant for the computation. Intuitively, the stuck theory is obtained as the symmetric and transitive closure of a relation that eliminates from a ρ-term all constraints whose matching problem is unsolvable independently of subsequent instantiations and reductions. This leads to a ρ-calculus that is suitable for the encoding of rewrite systems. We think that some properties of the encoded term rewrite system may be deduced from the development results on this version of the ρ-calculus.

The restrictions we assumed on ρ-patterns are certainly necessary for the confluence result. Indeed, as shown in [27, 10], non-algebraic and non-linear patters are sources of counter-examples to confluence. We conjecture that these restrictions, in particular linearity, can be weakened without loosing termination of developments. However, the treatment of non-linear calculi is delicate and reserve a deeper study. As pointed out in Section 4, we would need to change the definition of the underlined ρ-calculus by allowing to underline any abstraction application, even those corresponding to unsuccessful matching problems. This would lead to a less fine-grained control on redexes: an underlined ρ-term in normal form, for example, may still contain some underlinings.
On the other hand, linearity is most likely to be necessary to achieve a standardisation result for the ρ-calculus, towards which the FD and FD! theorems can be seen as a first step. We know from the two theorems on developments that the result of this kind of computations is unique and does not depend on the choice of a particular reduction. We still lack of information about the way to perform

the computation in order to reach this result, when it exists. This is definitely an interesting subject for future research. In particular, we are currently analysing the possibility of applying the axiomatic method presented in [15] to the ρ-calculus. It would be interesting to define a notion of standard reduction for the ρ-calculus by choosing a reduction order, *e.g.* leftmost outermost, which provides a terminating strategy for the calculus. Observe anyway that for the ρ-calculus with delayed matching constraints, this is not straight forward, since in a term of the form $[p \ll t_1]t_2$ the matching problem can be unsolvable at first, but some instantiation or reduction in t_1 can create a redex at the head position.

References

[1] H. Barendregt. *Lambda Calculus: its Syntax and Semantics*. North Holland, 1984.

[2] G. Barthe, H. Cirstea, C. Kirchner, and L. Liquori. Pure Patterns Type Systems. In *Principles of Programming Languages - POPL2003, New Orleans, USA*. ACM, January 2003.

[3] C. Bertolissi. *The graph rewriting calculus : properties and expressive capabilities*. Thèse de doctorat, Institut National Polytechnique de Lorraine - INPL, Oct 2005.

[4] A. Church. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5:56–68, 1941.

[5] A. Church and J. B. Rosser. Some properties of conversion. *Trans. Amer. Math. Soc.*, 40, 1936.

[6] H. Cirstea. *Calcul de réécriture : fondements et applications*. Thèse de Doctorat d'Université, Université Henri Poincaré - Nancy I, 2000.

[7] H. Cirstea, G. Faure, M. Fernández, I. Mackie, and F.-R. Sinot. New evaluation strategies for functional languages via the rho calculus. Currently submitted.

[8] H. Cirstea and C. Kirchner. The rewriting calculus — Part I *and* II. *Logic Journal of the Interest Group in Pure and Applied Logics*, 9(3):427–498, May 2001.

[9] H. Cirstea, C. Kirchner, and L. Liquori. Matching Power. In *Proc. of RTA*, volume 2051 of *LNCS*, pages 77–92. Springer-Verlag, 2001.

[10] H. Cirstea, C. Kirchner, and L. Liquori. Rewriting calculus with(out) types. In F. Gadducci and U. Montanari, editors, *Proceedings of the fourth workshop on rewriting logic and applications*, Pisa (Italy), Sept. 2002. Electronic Notes in Theoretical Computer Science.

[11] H. Cirstea, C. Kirchner, L. Liquori, and B. Wack. Rewrite strategies in the rewriting calculus. In B. Gramlich and S. Lucas, editors, *Proceedings of the Third International Workshop on Reduction Strategies in Rewriting and Programming* , Valencia, Spain, June 2003. Electronic Notes in Theoretical Computer Science.

[12] H. Cirstea, L. Liquori, and B. Wack. Rewriting Calculus with Fixpoints: Untyped and First-Order Systems. In *Proc. of Types, International Workshop on Types for Proof and Programs*, volume 3085 of *Lecture Notes in Computer Sciences*, pages 147–161. Springer Verlag, 2003.

[13] H. B. Curry and R. Feys. *Combinatory Logic*, volume 1. North Holland, 1958.

[14] M. Fernández, I. Mackie, and F.-R. Sinot. Interaction nets vs. the rho-calculus: Introducing bigraphical nets. In *Proceedings of EXPRESS'05, satellite workshop of Concur, San Francisco, USA, 2005*, Electronic Notes in Computer Science. Elsevier, 2005.

[15] G. Gonthier, J.-J. Levy, and P.-A. Mellies. An abstract standardisation theorem. In A. Scedrov, editor, *Proceedings of the Seventh Annual IEEE Symp. on Logic in Computer Science, LICS 1992*, pages 72–81. IEEE Computer Society Press, June 1992.

[16] J. Hindley. Reductions of residuals are finite. *Transactions of the American Mathematical Society*, 240:345–361, 1978.

[17] Horatiu Cirstea, Germain Faure, and Claude Kirchner. A rho-calculus of explicit constraint application. In *Proceedings of the 5th workshop on rewriting logic and applications*. Electronic Notes in Theoretical Computer Science, 2004.

[18] G. Huet. *Résolution d'equations dans les langages d'ordre 1,2, ...,ω*. Thèse de Doctorat d'Etat, Université de Paris 7 (France), 1976.

[19] J. W. Klop. *Combinatory Reduction Systems*. PhD thesis, CWI, 1980.

[20] L. Liquori and B. Serpette. iRho: an Imperative Rewriting Calculus. In *Proc. of PPDP, ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming*, pages 167–178. The ACM Press, 2004.

[21] P.-A. Melliès. *Description Abstraite des Systèmes de Réécriture*. PhD thesis, Université Paris 7, 1996.

[22] M. H. A. Newman. On theories with a combinatorial definition of equivalence. In *Annals of Math*, volume 43, pages 223–243, 1942.

[23] M. J. O'Donnell. *Computing in systems described by equations*. PhD thesis, Cornell, 1977.

[24] V. v. Oostrom. *Confluence for Abstract and Higher-Order Rewriting*. PhD thesis, Vrije Universiteit, Amsterdam, 1994.

[25] F. van Raamsdonk. *Confluence and Normalisation of Higher-Order Rewriting*. PhD thesis, University of Amsterdam, 1996.

[26] F. van Raamsdonk and P. Severi. On normalisation. In *148*, page 33. Centrum voor Wiskunde en Informatica (CWI), 30 1995.

[27] B. Wack. Klop counter example in the rho-calculus. Draft notes, LORIA, Nancy, 2003.