The Rewriting Calculus as a Combinatory Reduction System

Clara Bertolissi[1], Claude Kirchner[2]

[1]LIF-CMI, Université de Provence, Marseille, France
[2]INRIA & LORIA*, Nancy, France
`first.last@loria.fr, clara.bertolissi@lif.univ-mrs.fr`

Abstract. The last few years have seen the development of the *rewriting calculus* (also called rho-calculus or ρ-calculus) that uniformly integrates first-order term rewriting and λ-calculus. The combination of these two latter formalisms has been already handled either by enriching first-order rewriting with higher-order capabilities, like in the *Combinatory Reduction Systems* (CRS), or by adding to λ-calculus algebraic features.
In a previous work, the authors showed how the semantics of CRS can be expressed in terms of the ρ-calculus. The converse issue is adressed here: rewriting calculus derivations are simulated by Combinatory Reduction Systems derivations. As a consequence of this result, important properties, like standardisation, are deduced for the rewriting calculus.

Introduction

Lambda calculus and term rewriting are two foundational frameworks that had a deep influence on the development of computation and deduction. Starting from Klop's ground-breaking work on higher-order rewriting [15], and because of their complementarity, many frameworks have been designed with a view to integrate these two formalisms.

Introduced in the late nineties (see e.g. [7,8]), the rewriting calculus, also denoted ρ-calculus, combines uniformly the two paradigms and allows us to write $\lambda x.t$ to abstract, like in the λ-calculus, over the variable x, but also $\lambda p.t$ to abstract over an elaborated pattern p. Indeed, this last ρ-term is also written $p \twoheadrightarrow t$, emphasizing the rewriting aspect. This general abstract mechanism has been shown to be quite expressive and useful. For instance, it is well adapted to describe the semantics of imperative languages and object calculi [17,9] and the ρ-calculus has been used to model the execution of rewrite rules and strategies in rule-based languages like ELAN [10]. The logical aspects of the ρ-calculus are also quite appealing and provide the foundation for the design of a new class of proof assistants where computation and deduction can be adapted to the user's needs and understandings [22,4,12].

We are interested here in a better understanding of the behavior of the rewriting calculus by analyzing its derivation space. To this aim, we present an encoding of the ρ-calculus into CRSs, since for this kind of higher-order systems a

* UMR 7503 CNRS-INPL-INRIA-Nancy2-UHP

well-developed meta-theory already exists. Studies on the comparison between different higher-order formalisms has already been conducted between *Combinatory Reduction Systems* and *Higher-order Rewrite Systems* in [21]. Moreover, an encoding of CRSs into the rewriting calculus has already been proposed by the authors in [3]. This paper is concerned with the analysis of the converse relation, *i.e.* the study of derivations performed in rewriting calculus and their equivalent in higher-order rewriting, in particular in Combinatory Reduction Systems. We define a translation of the components of the ρ-calculus into the analogous notions in a CRS. Using this translation, we show that every derivation of a ρ-term has a corresponding derivation in the CRS and we prove the soundness and completeness of this encoding. We conclude by deriving some important properties concerning rewriting calculus reductions, as confluence, finiteness of developments and standardization, by using the well-known corresponding results in the CRSs.

The paper is structured as follows: in Section 1 we briefly present the ρ-calculus through its components. Section 2 provides a description of CRSs and some examples. In Section 3 we present a translation from ρ-terms and evaluation rules into CRS-terms and CRS-rewrite rules respectively, and we prove the completeness and soundness of the translation. Section 4 concludes the paper with some additional remarks and perspectives.

1 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].

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 $f, g, \ldots$ of fixed arity range over the infinite set $\mathcal{F}$. 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 "$\lambda\_.\_$" (abstraction operator), and "$\_ \wr \_$" (structure operator), and the (hidden) application operator. The set of ρ-terms is then defined as follows:

$$\begin{aligned} \mathcal{T} &::= \mathcal{X} \mid \mathcal{F} \mid \lambda\mathcal{P}.\mathcal{T} \mid \mathcal{T}\,\mathcal{T} \mid \mathcal{T} \wr \mathcal{T} \\ \mathcal{P} &::= \mathcal{X} \mid \mathcal{F} \mid \mathcal{F}\,\mathcal{P} \ldots \mathcal{P} \end{aligned}$$

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 and we usually denote a term of the form $(\ldots((f\ t_1)\ t_2)\ldots)\ t_n$ with $f \in \mathcal{K}$ of arity n by $f(t_1, t_2, \ldots, t_n)$. A *linear* pattern is a pattern where every variable occurs at most once. In the rest of the paper we will consider only *well-formed* terms, *i.e.* terms where functional symbols are provided with the correct number of arguments, according to their arity. Moreover, we will restrict to linear patterns, a standard restriction that will allow us to reuse the properties of CRSsthat are indeed, in general, assuming left linearity.

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 "$\lambda\_.\_$" which is, in turn, of higher priority than the "$\_ \wr \_$".

Let $\Box$ be a fresh symbol, called a *hole*. A term with one or more occurrences of $\Box$ is called a *context* and denoted by $\mathsf{Ctx}_{\lceil\ \rceil}$. An *algebraic* context is an algebraic term with one or more occurrences of $\Box$. A *ground* context is a context containing no variables. The term obtained by replacing from left to right in a context $\mathsf{Ctx}_{\lceil\ \rceil}$ the n holes $\Box$ by the terms $t_1, \ldots, t_n$, $n \geq 1$, is denoted by $\mathsf{Ctx}_{\lceil t_1,\ldots,t_n \rceil}$.

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 1 (Free variables). *The set of free variables of a ρ-term t, denoted $\mathcal{FV}(t)$ is inductively defined as follows:*

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

The set $\mathcal{BV}$ of bound variables of a term is the complementary of the set of free variables w.r.t. the set of variables of the respective term. A term is called closed *if all its variables are bound.*

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 consisting of the two rules $f(a) \to a$ and $f(a) \to b$.

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

Definition 2 (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 $\mathcal{D}om(\sigma) = \{x_1, \ldots, x_m\}$. Applying a substitution σ to a term t, denoted by $\sigma(t)$ or $t\sigma$, is defined as follows:*

$$\begin{array}{llll} \sigma(f) & = f & \sigma(\lambda p.t) & = \lambda p.\sigma(t) \\ \sigma(x_i) & = \begin{cases} t_i & \textit{if } x_i \in Dom(\sigma) \\ x_i & \textit{otherwise} \end{cases} & \begin{array}{l} \sigma(t_1\, t_2) \\ \sigma(t_1 \wr t_2) \end{array} & \begin{array}{l} = \sigma(t_1)\ \sigma(t_2) \\ = \sigma(t_1) \wr \sigma(t_2) \end{array} \end{array}$$

We point out that we work modulo α-*convention*: when applying a substitution to an abstraction, we know that the free variables of the corresponding abstracted pattern do not belong to the domain of the substitution.

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 have at most one solution and are known to be decidable [13].

Definition 3 (Syntactic matching). *A (syntactic) matching problem is a formula of the form $p \ll t$, where p is a pattern and t is a term. A substitution σ is solution of the matching problem $p \ll t$, denoted by $Sol(p \ll t)$, if $\sigma(p) \equiv t$.*

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

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

The (ρ)-rule can be applied if (and only if) a substitution of the matching problem $p \ll t_3$ 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. 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 $\mapsto_{\rho\delta}$.

Example 2 (Reductions). We consider the ρ-terms of Example 1 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 $Sol(plus(x,0) \ll plus(n,0)) = \{n/x\}$, we have the reduction $(\lambda plus(x,0).x)\,plus(n,0) \mapsto_\rho n$;
3. $(\lambda f(a).a \wr \lambda f(a).b)\ f(a) \mapsto_\delta (\lambda f(a).a)\ f(a) \wr (\lambda f(a).b)\ f(a) \mapsto\!\!\!\!\to_\rho a \wr b$ is a ρ-reduction capturing the non-determinism of first-order term rewriting.

2 The Combinatory Reduction Systems

The *Combinatory Reduction Systems* (CRSs), introduced by J.W. Klop in 1980 [15], are a generalization of first-order term rewrite systems with a mechanism of bound variables like in the λ-calculus. The definitions of this section are based on the presentation of CRSs given in [16].

In what follows the symbols $A, L, R, \ldots$ range over the set $\mathcal{MT}$ of so called *meta-terms*, $t, u, \ldots$ range over the set $\mathcal{T}_{\text{CRS}}$ of *terms*, $x, y, z, \ldots$ range over the set

$\mathcal{X}$ of *variables*, $X, Z \ldots$ range over the set $\mathcal{Z}$ of *meta-variables* of fixed arity and $f, g, \ldots$ range over the set $\mathcal{F}$ of *functional symbols* of fixed arity. We denote by $\mathcal{F}_i \subset \mathcal{F}$ ($\mathcal{Z}_i \subset \mathcal{Z}$) the subset of symbols (meta-variables) of arity i. All symbols can be indexed. The set of CRS-meta-terms is defined as follows:

$$\mathcal{MT} ::= \mathcal{X} \mid \mathcal{F}_n(\mathcal{MT}_1, \ldots, \mathcal{MT}_n) \mid \mathcal{Z}_n(\mathcal{MT}_1, \ldots, \mathcal{MT}_n) \mid [\mathcal{X}]\mathcal{MT}$$

The set $\mathcal{T}_{\text{CRS}} \subset \mathcal{MT}$ of CRS-terms is composed of all the meta-terms without meta-variables. We should point out that all meta-terms are *well-formed*, *i.e.* the functional symbols and meta-variables take exactly as many arguments as their arity. A CRS context is defined similarly as in the ρ-calculus.

The operator $[\_]\_$ denotes an abstraction similar to the abstraction of the λ-calculus such that in $[x]t$ the variable x is bound in t. In a meta-term of the form $[x]A$ we call A the scope of $[x]$. A variable x occurs *free* in a meta-term if it is not in the scope of an occurrence of $[x]$. A variable x occurs *bound* otherwise. The set of free variables of a meta-term A is written $\mathcal{FV}(A)$.

Meta-variables (in the CRS rewrite rules defined below) behave as (free) variables of first-order rewrite systems. The set of meta-variables of a meta-term A is written $\mathcal{MV}(A)$. As for the ρ-calculus, we work modulo the α-conversion.

Example 3 (Terms and Metaterms). Some examples of terms and metaterms:

- $f([x]g(x, a)) \in \mathcal{T}_{\text{CRS}}$ with $f \in \mathcal{F}_1$, $g \in \mathcal{F}_2$, $a \in \mathcal{F}_0$.
- $Z_1(Z_2) \in \mathcal{MT}$ with $Z_1 \in \mathcal{Z}_1$, $Z_2 \in \mathcal{Z}_0$.
- $f([x]Z(x, y)) \in \mathcal{MT}$ with $f \in \mathcal{F}_1$, $Z \in \mathcal{Z}_2$.

The application of substitutions is defined at the meta-level of the calculus and uses $\underline{\lambda}$-calculus as meta-language (underlined just for distinguishing it from classical λ-calculus). Unintended bindings of variables by the $\underline{\lambda}$-abstractor operator are avoided using α-conversion. To simplify the notation we denote $\underline{\lambda}x_1 \ldots \underline{\lambda}x_n.t$ by $\underline{\lambda}x_1 \ldots x_n.t$. The reduction of $\underline{\lambda}$-redexes is performed by the $\underline{\beta}$-rule of the $\underline{\lambda}$-calculus. The $\underline{\beta}$-normal form of a term t is denoted by $\downarrow_{\underline{\beta}}$. We should point out that a CRS-(meta)term is necessarily in $\underline{\beta}$-normal form.

Performing a substitution in a CRS corresponds to applying an *assignment* (and consequently a set of substitutes) to a CRS-meta-term.

Definition 4 (Substitute, assignment).
An n-ary substitute *is an expression of the form* $\xi = \underline{\lambda}x_1 \ldots x_n.u$ *where* $x_1, \ldots, x_n$ *are distinct variables and* u *is a* CRS*-term. Its application to an n-tuple of* CRS*-terms* $t_1, \ldots, t_n$ *yields the simultaneous substitution of* x_i *by* t_i *in* u, $i = 1 \ldots n$, *denoted* $(\underline{\lambda}x_1 \ldots x_n.u)(t_1, \ldots, t_n)\downarrow_{\underline{\beta}} = u\{x_1/t_1, \ldots, x_n/t_n\}$.

An assignment $\sigma = \{(Z_1, \xi_1), \ldots, (Z_n, \xi_n)\}$, *is a finite set of pairs (metavariable, substitute) such that arity(Z_i) = arity(ξ_i)* $\forall i \in \{1, \ldots, n\}$ *(Dom(σ) = $\{Z_1, \ldots, Z_m\}$). The application of an assignment σ to a* CRS*-meta-term A, denoted $\sigma(A)$ or $A\sigma$, is inductively defined by:*

$$\begin{array}{llll}
\sigma(x) = x & & \sigma([x]A) & = [x]\sigma(A) \\
\sigma(Z_i) = \xi_i & \textit{if } (Z_i, \xi_i) \in \sigma & \sigma(f(A_1, \ldots, A_n) & = f(\sigma(A_1), \ldots, \sigma(A_n)) \\
\sigma(Z_i) = Z_i & \textit{if } Z_i \notin Dom(\sigma) & \sigma(Z_i(A_1, \ldots, A_n)) & = \sigma(Z_i)(\sigma(A_1), \ldots, \sigma(A_n))\downarrow_{\underline{\beta}}
\end{array}$$

Notice that the assignments have no effect on variables since they can instantiate only meta-variables. Since we work modulo the α-convention, unintended bindings of free variables are avoided by renaming bound variables.

A CRS rewrite rule is a pair of metaterms. We consider as left-hand side of the rules only the CRS-meta-terms satisfying the CRS-pattern definition:

Definition 5 (CRS-pattern). *A* CRS*-metaterm P is a* CRS-pattern *if any of its metavariables Z appears in a sub-metaterm of P of the form $Z(x_1, \ldots, x_n)$ where the variables $x_1, \ldots, x_n$, $n \geq 0$, are distinct and all bound in P.*

In this paper we only consider rewrite rules with CRS-patterns as left-hand sides and satisfying the usual conditions imposed in first-order rewriting:

Definition 6 (Rewrite rules). *A set of* CRS *rewrite rules consists of rules of the form $L \to R$ satisfying the following conditions:*

- *L and R are closed metaterms ($\mathcal{FV}(L) = \mathcal{FV}(R) = \emptyset$);*
- *L has the form $f(A_1, \ldots, A_n)$ with $A_1, \ldots, A_n$ metaterms and $f \in \mathcal{F}_n$;*
- *$\mathcal{MV}(L) \supseteq \mathcal{MV}(R)$;*
- *L is a* CRS*-pattern.*

The last condition ensures the decidability and the uniqueness of the solution of the matching inherent to the application of the CRS-rules [3]. Moreover, the additional condition of linearity can be required for the metaterm L, meaning that L contains no multiple occurrences of the same metavariable.

Example 4 (β-rule in CRS*s).* The β-rule of λ-calculus $(\lambda x.t)u \to_\beta t\{x/u\}$ can be expressed as the rewrite rule: $App(Ab([x]Z(x)), Z_1) \to Z(Z_1)$. In this rule, called β_{CRS} in this paper, $App \in \mathcal{F}_2$ and $Ab \in \mathcal{F}_1$ are the encodings for the λ-calculus application and abstraction operators respectively.

Given a rewrite rule $L \to R$ and a substitution σ, we have $\sigma(L) \to_{L \to R} \sigma(R)$ if $\sigma(L), \sigma(R) \in \mathcal{T}_{\text{CRS}}$. The term $\sigma(L)$ is called a *redex*. The left-hand side and the right-hand side of a CRS rewrite rule are metaterms, but the rewrite relation induced by the rule is a relation on terms.

Given a set of CRS rewrite rules $\mathcal{R}$, the corresponding one-step relation $\longmapsto_{\mathcal{R}}$ (denoted also $\longmapsto_{L \to R}$ if we want to specify the applied rule) is the context closure of the relation induced (as above) by the rules in $\mathcal{R}$. The multi-step evaluation $\mapsto\!\!\!\!\to_{\mathcal{R}}$ is defined as the reflexive and transitive closure of $\longmapsto_{\mathcal{R}}$.

Example 5. Let us consider the CRS-term $f(App(Ab([x]f(x)), a))$. We apply to the sub-term $App(Ab([x]f(x)), a)$ the β_{CRS} rule (Example 4) using the assignment $\sigma = \{(Z, \underline{\lambda} y.f\ y), (Z_1, a)\}$. As result, we obtain the instantiation by σ of the right-hand side R of the rule β_{CRS}: $\sigma(R) = \sigma(Z(Z_1)) = (\sigma(Z))(\sigma(Z_1)) = (\underline{\lambda} y.f\ y)(a)\downarrow_{\underline{\beta}} = f(a)$. Therefore we have $App(Ab([x]f(x)), a) \longmapsto_{\beta_{CRS}} f(a)$ and thus $f(App(Ab([x]f(x)), a)) \longmapsto_{\beta_{CRS}} f(f(a))$.

One can notice that there are two binding mechanisms in the formalism presented in this section. The first one is explicit in the syntax and denoted $[x]t$. The second one that is implicit concerns the metavariables and comes from the rewriting mechanism.

CRSs as defined so far are quite general and do not satisfy important properties like, for example, confluence. If we restrict to the class of orthogonal CRSs, confluence and other interesting properties are satisfied [15].

Definition 7 (Orthogonality). *Let $\mathcal{R} = \{L_i \to R_i | i \in I\}$ be a set of* CRS *rewrite rules.*

1. *$\mathcal{R}$ is* non-overlapping *if the following holds:*
   Let r_i be the redex $\sigma(L_i)$ and let $Z_1, \dots Z_n$ be all the distinct metavariables of the metaterm L_i. Then if r_i contains another redex $r_j = \sigma(L_j)$, this redex r_j must be already present in $\sigma(Z_p(x_{i_1} \dots x_{i_{k_p}}))$, for some subterm $Z_p(x_{i_1} \dots x_{i_{k_p}})$ of L_i.
2. *$\mathcal{R}$ is* left-linear *if all the metaterms L_i are left-linear.*
3. *$\mathcal{R}$ is* orthogonal *if it is non-overlapping and left-linear.*
4. *A* CRS *is orthogonal if it has an orthogonal set of rewrite rules.*

Similarly to the λ-calculus, CRSs can be equipped with simple types. All types are generated from base types $\tau_0, \dots, \tau_n$ in the usual way. Variables and constants have a (unique) base type, an abstraction $[x]A$ has type $[x]A : \tau_0 \to \tau_1$ if $x : \tau_0$ and $A : \tau_1$, a metaterm $g(A_1, \dots, A_n)$ with $g \in \mathcal{F}_n$ has type τ_0 if $g : \tau_1 \dots \to \tau_n \to \tau_0$ and $A_i : \tau_i$, for $i = 1 \dots n$. Similarly for a metaterm $Z(A_1, \dots, A_n)$.

3 Translating Rewriting Calculus into CRS

The two systems introduced in the previous sections can be seen as two formats of higher-order rewriting which, in spite of their differences in the presentation, use similar mechanisms for performing computations.

We propose in this section an analysis of the two calculi in order to point out their similarities. In particular, we define a translation function from the ρ-calculus to simply-typed CRS and we prove the completeness and soundness of the translation. The results are obtained using an equivalence between the rewrite relations of the two systems, making it possible to transfer the properties holding for one system to the other, as discussed in Section 3.4.

3.1 The translation

In the following we choose a CRS having as variables the set of variables $\mathcal{X}$ of the ρ-calculus and having as functional symbols the set of constants $\mathcal{K}$ of the ρ-calculus plus four distinguished symbols: the binary symbols $App, Dis, rule$ and the unary symbol Ab. The types of the CRS-terms are built from only one base type that we denote τ.

Definition 8 (Translation of terms).

The translation, denoted $\overline{t}$, of a ρ-term t into a simply typed CRS*-term, is defined as follows:*

$$\begin{array}{ll}
\overline{x} & = x \textit{ with } x \textit{ of type } \tau \\
\overline{f(t_1,\ldots,t_n)} & = f(\overline{t_1},\ldots,\overline{t_n}) \textit{ with } f:\tau\to\ldots\to\tau \;(n \textit{ type arrows}) \\
\overline{t_1\, t_2} & = App(\overline{t_1},\overline{t_2}) \textit{ with } App:\tau\to\tau\to\tau \\
\overline{t_1 \wr t_2} & = Dis(\overline{t_1},\overline{t_2}) \textit{ with } Dis:\tau\to\tau\to\tau \\
\overline{\lambda p.t} & = Ab([x_1]\ldots[x_n].rule(\overline{p},\overline{t})) \textit{ where } \{x_1\ldots x_1\}=\mathcal{FV}(p) \\
& \quad \textit{with } Ab:(\tau\to\ldots\to\tau)\to(\tau\to\tau)\to\tau \\
& \quad \textit{and } rule:\tau\to\tau\to(\tau\to\tau)
\end{array}$$

Given a context $\mathsf{Ctx}_{\lceil\ \rceil}$, a hole $\Box$ is of base type.

The ρ-calculus functional application is translated into a functional symbol with associated arguments and corresponding type. The application and the structure operator are translated by the two special symbols *App* and *Dis* of arity two. Pattern abstractions need a more subtle translation. Since the CRS operator $[\_]\_$ abstracts on single variables, we use an intermediary distinguished symbol *rule* which takes as arguments the pattern and the right-hand side of the rule, and then we abstract on the variables of the pattern. Since we want the result to have a base type, we enclose the resulting CRS-term with a symbol *Ab* meant to collapse a functional type.

We can immediately notice that the CRS-terms obtained as translation of some ρ-terms have good structural properties.

Proposition 1 (Properties of the translation).

i) For any well-formed ρ-term t, we have $\overline{t}:\tau$.

ii) For any subterm $s:\tau$ of a CRS*-term $\overline{t}$, there exists a ρ-term s' such that $\overline{s'}=s$.*

Proof. By structural induction on the ρ-term t and the CRS-term $\overline{t}$, respectively.

These properties will be useful later on for proving the soundness of the translation.

The set of rewrite rules of the CRS is obtained by translating the evaluation rules of the ρ-calculus.

Definition 9 (Evaluation rules encoding). *Let $p_{\lceil\ \rceil}$ denote a ground algebraic* CRS *context with n holes. The translation of the evaluation rules of the ρ-calculus into an (infinite) set $\mathcal{R}$ of* CRS *rewrite rules (schematic in $p_{\lceil\ \rceil}$) is then defined as follows:*

$$\begin{array}{ll}
(\rho_p) & App(Ab([x_1]\ldots[x_n].rule(p_{\lceil x_1,\ldots,x_n\rceil}, Z(x_1,\ldots,x_n))), p_{\lceil Z_1,\ldots,Z_n\rceil}) \\
& \quad \to \quad Z(Z_1,\ldots,Z_n)
\end{array}$$

$$(\delta_C)\ App(Dis(Z_1,Z_2),Z_3) \quad \to \quad Dis(App(Z_1,Z_3),App(Z_2,Z_3))$$

The δ_C-rule is a direct encoding of the (δ) evaluation rule of the rewriting calculus, where the structure operator and the application operator have been replaced by the corresponding functional symbols in the CRS. The first CRS-rewrite rule, the ρ_p-rule, is used to reduce the CRS-redexes corresponding to an abstraction application in the ρ-calculus. To any ρ-reduction $t_0 = (\lambda p.t_1)\ t_3 \mapsto_{\overline{\rho}} \dots$ we associate a CRS-reduction $\overline{t_0} \mapsto_{\rho_p} \dots$. The context $p_{\lceil\ \rceil}$ in the corresponding ρ_p-rule is obtained by translating the ρ-pattern p into the CRS and replacing its n variables by n holes. Moreover, the translation implicitly defines a relation between the free variables of the ρ-pattern p, say $y_1, \dots, y_n$, and the metavariables $Z_1, \dots, Z_n$ in the ρ_p-rule. We can formalise this relation using an injective function $\zeta : \mathcal{X} \mapsto \mathcal{Z}$ such that $\zeta(y_i) = Z_i \in \mathcal{Z}_0$ for all $i = 1 \dots n$.

The side condition of the rule (ρ), *i.e.* $\sigma = Sol(p \prec\!\!\prec t_3)$, is encoded directly inside the CRS-rule ρ_p, by choosing the same structure of the context $p_{\lceil\ \rceil}$ in the subterm $rule(p_{\lceil x_1,\dots,x_n \rceil}, \dots)$, encoding the ρ-abstraction, and in the metaterm $p_{\lceil Z_1,\dots,Z_n \rceil}$ to which the abstraction is applied. This corresponds to the encoding of the rule (ρ) with syntactic matching.

Observe that the rule schema ρ_p represents an infinity of rewrite rules, one for each $p_{\lceil\ \rceil}$. In principle, we suppose to have a rule ρ_p for each algebraic linear ρ-pattern p. In practice, for the properties we are interested in, we will only need a finite number of rules ρ_p, as discussed in Section 3.4.

In the following we will often denote the left-hand side of the rules δ_C and ρ_p by L_C and L_p, respectively. It is not difficult to see that L_C and L_p are CRS-patterns:

Proposition 2. *The metaterms L_C and L_p of the rewrite rules δ_C and ρ_p are* CRS*-patterns, i.e. verify Definition 5, for any context $p_{\lceil\ \rceil}$.*

We can remark that this would not be the case without the assumption of linearity on ρ-patterns. As a consequence of L_C and L_p being CRS-patterns, matching in the obtained CRS is decidable and unitary [3].

Example 6. In the rule ρ_p, consider the empty context for $p_{\lceil\ \rceil}$ and $n = 1$ (so that we abstract on a single variable). We obtain in this way the following encoding in the CRS of the β-rule of the λ-calculus:

$$App(Ab([x].rule(x, Z(x))), Z_1) \quad \to \quad Z(Z_1)$$

which is slightly different from the usual encoding, presented in Example 4.

Example 7. Consider the ρ-term $t = (\lambda g(x, y).f(x))\ g(a, b)$. In the ρ-calculus we have the reduction $(\lambda g(x, y).f(x))\ g(a, b) \ \mapsto_{\overline{\rho}}\ f(a)$.

The translation of the term t into a CRS-term is $\overline{t} = App(Ab([x][y].rule(g(x, y), f(x))), g(a, b))$. We can apply to $\overline{t}$ the CRS-rewrite rule $\rho_{g(x_1,x_2)}$: $App(Ab([x_1][x_2].rule(g(x_1, x_2), Z(x_1, x_2))), g(Z_1, Z_2)) \ \to\ Z(Z_1, Z_2)$ using the assignment $\sigma = \{(Z/\underline{\lambda} z_1 z_2.f(z_1), Z_1/a, Z_2/b\}$. We obtain

$$\begin{aligned} &App(Ab([x][y].rule(g(x, y), f(x))), g(a)) \\ \mapsto_{\rho_P}\ &\sigma(Z(Z_1, Z_2)) = \sigma(Z)(\sigma(Z_1), \sigma(Z_2)) = (\underline{\lambda} z_1 z_2.f(z_1))(a, b){\downarrow_{\underline{\beta}}} = f(a) \end{aligned}$$

3.2 Completeness of the translation

The aim of this section is to show that for every ρ-reduction there exists a corresponding CRS-reduction. We start by proving a lemma expressing the interaction between the formation of contexts and substitutions and the translation. We then show the simulation of one-step ρ-reductions in the CRS and we conclude the section by a theorem stating the completeness of the translation.

Lemma 1 (Context stability). *Let $s, s_1, \ldots s_n$ be ρ-terms and $\mathsf{Ctx}_{\lceil\ \rceil}$ be a context. Then we have*

i) $\overline{\mathsf{Ctx}_{\lceil s \rceil}} = \overline{\mathsf{Ctx}}_{\lceil \overline{s} \rceil}$
ii) $\overline{\mathsf{Ctx}_{\lceil s_1,\ldots,s_n \rceil}} = \overline{\mathsf{Ctx}}_{\lceil \overline{s_1},\ldots,\overline{s_n} \rceil}$
iii) $\overline{s\{x_1/s_1,\ldots,x_n/s_n\}} = \overline{s}\{x_1/\overline{s_1},\ldots,x_n/\overline{s_n}\}$

Proof. By induction on the structure of the context $\mathsf{Ctx}_{\lceil\ \rceil}$.

The previous lemma is used to show that rewrite steps are naturally preserved by the translation.

Lemma 2 (One-step simulation). *Given a ρ-term t such that $t \mapsto_{\rho\delta} t_1$, then in the corresponding* CRS *we have $\overline{t} \mapsto_{\mathcal{R}} \overline{t_1}$.*

Proof. Without loss of generality, we suppose the redex being at the head position in the ρ-term t.

- $t \mapsto_{\delta} t_1$ then $t = (t_1 \wr t_2)\, t_3$ and $t' = t_1\, t_3 \wr t_2\, t_3$. We have $\overline{t} = App(Dis(\overline{t_1}, \overline{t_2}), \overline{t_3})$, we apply the rule (δ_C) using the assignment $\sigma = \{Z_1/\overline{t_1}, Z_2/\overline{t_2}, Z_3/\overline{t_3}\}$ and we obtain $\sigma(Dis(App(Z_1, Z_3), App(Z_2, Z_3))) = Dis(App(\overline{t_1}, \overline{t_3}), App(\overline{t_2}, \overline{t_3})) = \overline{t_1}$.
- $t \mapsto_{\rho} t_1$ then $t = (\lambda p.v)\ u$ and $t_1 = \sigma(v)$ with $\mathcal{FV}(p) = \{x_1, \ldots, x_n\}$ and $\sigma = Sol(p \ll u) = \{x_1/u_1, \ldots, x_n/u_n\}$.
In the CRS, we have $\overline{t} = App(Ab([x_1]\ldots[x_n].rule(\overline{p}, \overline{v})), \overline{u})$. We can apply the corresponding CRS-rewrite rule (ρ_p) using the assignment $\sigma' = \{Z/\underline{\lambda} z_1 \ldots z_n.\overline{v}', Z_1/\overline{u_1}, \ldots, Z_n/\overline{u_n}\}$ where $\overline{v}'$ is the term $\overline{v}$ to which a renaming of the variables x_i into z_i, for $i = 1 \ldots n$, has been applied. By applying the converse renaming and by Lemma 1, we obtain $\sigma'(Z(Z_1, \ldots, Z_n)) = (\underline{\lambda} z_1 \ldots z_n.\overline{v}')(\overline{u_1}, \ldots, \overline{u_n}) \downarrow_{\underline{\beta}} = \overline{v}'\{z_1/\overline{u_1}, \ldots, z_n/\overline{u_n}\}$ $= \overline{v}\{x_1/\overline{u_1}, \ldots, x_n/\overline{u_n}\} = \overline{v\{x_1/u_1, \ldots, x_n/u_n\}} = \overline{\sigma(v)} = \overline{t_1}$.

The generalisation to derivations of arbitrary length follows easily:

Theorem 1 (Completeness). *Given a ρ-term t such that $t \mapsto\!\!\!\rightarrow_{\rho\delta} t_n$, then in the corresponding* CRS *we have $\overline{t} \mapsto\!\!\!\rightarrow_{\mathcal{R}} \overline{t_n}$.*

Proof. By induction on the length of the reduction, using Lemma 2.

We can notice that there is a one-to-one correspondence in the derivations, *i.e.* for every step in the rewriting calculus a corresponding step is performed in the CRS. The substitution application is performed at the meta-level in both calculi, by underlined beta-reductions in the CRS and by simultaneous variable substitutions in the ρ-calculus, and therefore does not affect the length of the reductions.

3.3 Soundness of the translation

Completeness proves that for every rewrite step in ρ-calculus, a rewrite step in the associated CRS can be performed. We will show now that a rewrite step in a translated term must originate from a rewrite step in the ρ-calculus. We state first a precise relation between a CRS assignment and the corresponding ρ-calculus substitution.

Lemma 3. *Let $t = (\lambda p.v)\ u$ be a ρ-term with $\mathcal{FV}(p) = \{y_1, \dots y_n\}$ and L_p be the left hand side of the* CRS*-rewrite rule ρ_p with $Z_i = \zeta(y_i)$ for all $i = 1 \dots n$. Let σ be an assignment such that $\sigma(L_p) = \overline{t}$. Then*

1. *the assignment σ is of the form $\sigma = \{Z/\underline{\lambda} z_1 \dots z_n.\overline{v}, Z_1/\overline{u_1}, \dots, Z_n/\overline{u_n}\}$.*
2. *in the ρ-calculus the substitution $\sigma' = \{y_1/u_1, \dots, y_n/u_n\}$ is such that $\sigma'(p) = u$.*

Proof. 1. We show that the given assignment σ is a solution of the matching of the CRS metaterm L_p to the CRS-term $\overline{t}$. Since the solution of a CRS pattern matching problem is unique [3], this concludes the proof.

2. By Lemma 1 and the injectivity of the translation function.

We can show now that a rewrite step in the translation of the ρ-calculus is related with a rewrite step in the ρ-calculus itself.

Lemma 4. *If $\overline{t} \mapsto_{\mathcal{R}} t_1$ in the* CRS*, then we have $t \mapsto_{\rho\delta} t'$ with $\overline{t'} = t_1$.*

Proof. By induction on the depth of the redex position in the term $\overline{t}$.

Base case: the redex is at the head position in the term $\overline{t}$. Then we have:

- $\overline{t} \mapsto_{\delta_C} t_1$ with $\overline{t} = App(Dis(\overline{t_1}, \overline{t_2}), \overline{t_3})$ and $t_1 = Dis(App(\overline{t_1}, \overline{t_3}), App(\overline{t_2}, \overline{t_3}))$ using the assignment $\sigma = \{Z_1/\overline{t_1}, Z_2/\overline{t_2}, Z_3/\overline{t_3}\}$. In the ρ-calculus we have $t = (t_1 \wr t_2)\, t_3 \mapsto_\delta t' = t_1\, t_3 \wr t_2\, t_3$ and thus it is easy to see that $\overline{t'} = t_1$.
- $\overline{t} \mapsto_{\rho_P} t_1$. We have $\overline{t} = App(Ab([x_1] \dots [x_n].rule(\overline{p}, \overline{v})), \overline{u})$ which reduces to $t_1 = \overline{v}'\{z_1/\overline{u_1}, \dots, z_n/\overline{u_n}\}$ using the assignment $\overline{\sigma} = \{Z/\underline{\lambda} z_1 \dots z_n.\overline{v}', Z_1/\overline{u_1}, \dots, Z_n/\overline{u_n}\}$, where $\overline{v}'$ is the term $\overline{v}$ to which a renaming of the variables x_i into z_i, for $i = 1 \dots n$, has been applied. In the ρ-calculus we have $t = (\lambda p.v)\ u$ with $\mathcal{FV}(p) = \{x_1, \dots x_n\}$. By Lemma 3 there exists a substitution $\sigma = \{x_1/u_1, \dots, x_n/u_n\}$ solution of the matching problem $p \ll u$. Thus we obtain $t \mapsto_\rho t' = \sigma(v) = v\{x_1/u_1, \dots, x_n/u_n\}$. Using the converse renaming of z_i into x_i, for $i = 1 \dots n$, by Lemma 1, we conclude $\overline{t'} = t_1$.

Induction: If the redex is not at the head position in the term $\overline{t}$. Then we have $\overline{t} = \mathsf{Ctx}_{\lceil \sigma(L) \rceil} \mapsto_R \mathsf{Ctx}_{\lceil \sigma(R) \rceil}$, for some context $\mathsf{Ctx}_{\lceil\ \rceil}$. Since $\overline{t}$ and $\sigma(L)$ are of base type by Proposition 1 we have a ρ-context $\mathsf{Ctx}'_{\lceil\ \rceil}$ such that $\overline{\mathsf{Ctx}'_{\lceil\ \rceil}} = \mathsf{Ctx}_{\lceil\ \rceil}$ and a ρ-term s such that $\overline{s} = \sigma(L)$. Thus, by induction $t = \mathsf{Ctx}'_{\lceil s \rceil} \mapsto_{\rho\delta} \mathsf{Ctx}'_{\lceil s' \rceil} = t'$ with $\overline{s'} = \sigma(R)$. Hence, using Lemma 1, $\overline{t'} = \overline{\mathsf{Ctx}'_{\lceil s' \rceil}} = \overline{\mathsf{Ctx}'}_{\lceil \overline{s'} \rceil} = \mathsf{Ctx}_{\lceil \sigma(R) \rceil} = t_1$

The previous lemma is generalised to arbitrary rewrite sequences in the following theorem.

Theorem 2 (Soundness). *If* $\overline{t} \mapsto\!\!\!\!\twoheadrightarrow_{\mathcal{R}} t_n$ *in the* CRS*, then we have* $t \mapsto\!\!\!\!\twoheadrightarrow_{\rho\delta} t'$ *with* $\overline{t'} = t_n$.

Proof. By induction on the length of the reduction, using Lemma 4.

Soundness and completeness say that the fact that a term rewrites to another is preserved and reflected in the two systems. This can be seen as a measure of the neatness of the translation. Indeed, other important properties, as discussed in the following section, can be reflected from the CRS into the ρ-calculus.

3.4 Properties

The obtained results allow us to deduce important properties for the rewriting calculus via the properties of the corresponding CRS, avoiding the development of *ad hoc* proofs for the ρ-calculus.

In particular, we are interested in properties as confluence, finiteness of developments and standardisation, that analyse the derivation space starting from an initial ρ-term. In the version of the rewriting calculus considered here (and contrarily to the dynamic patterns used in [1]), all variables in a ρ-pattern p are bound, therefore no new patterns can be created during the reduction. Therefore, we can consider a CRS having a finite number of rewrite rules, that is the rewrite rule (δ_C) and for any pattern p present in the initial ρ-term, a corresponding rule (ρ_P).

First of all, we can notice that any CRS obtained from the translation belongs to the class of orthogonal CRSs, since it is left-linear and non-overlapping:

Lemma 5 (Orthogonality). *The* CRS *obtained as result of the translation of the* ρ*-calculus is an orthogonal* CRS*.*

Proof. First, it is clear that the CRS-patterns L_p and L_C are linear, for any context $p_{\lceil\ \rceil}$. We show next that the rules (δ_C) and (ρ_P) are non-overlapping.

Let t, t' be two CRS-terms. We show that if there exists an assignment σ such that $\sigma(L_p) = t$ and $t \mapsto\!\!\!\!\twoheadrightarrow_{\mathcal{R}} t'$, then there exists an assignment σ' such that $\sigma'(L_p) = t'$. Suppose $t = \mathsf{Ctx}_{\lceil \sigma_2(L) \rceil} \mapsto\!\!\!\!\twoheadrightarrow_{\mathcal{R}} \mathsf{Ctx}_{\lceil \sigma_2(R) \rceil} = t'$. The fact that the assignment σ exists implies that the occurrence of the redex $\sigma_2(L)$ in t, say ω, corresponds to a metavariables position in L_p, say the position of the metavariable Z_i. Thus σ of the form $\{Z/s_0, Z_1/s_1, \ldots, Z_i/\sigma_2(L), \ldots, Z_n/s_n\}$ is such that $\sigma(L_p) = t$. The contraction of the redex $\sigma_2(L)$ in t affects only the subterm of t headed in ω, therefore the substitution $\sigma' = \{Z/s_0, Z_1/s_1, \ldots, Z_i/\sigma_2(R), \ldots, Z_n/s_n\}$ is such that $\sigma'(L_p) = t'$. A similar reasoning can be done for the left-hand side L_C of the CRS rewrite rule δ_C.

As a consequence of orthogonality, we can deduce immediately the confluence property for the rewriting calculus. This property has already been proved for various versions of the calculus, in particular in [5]. A confluence proof is also available for the typed version of the left linear calculus called *PPTS* [1]. Here we obtain the confluence property for the ρ-calculus exploiting the well-known results for orthogonal CRSs:

Corollary 1 (Confluence). *The ρ-calculus with linear algebraic patterns and syntactic matching is confluent.*

Proof. By the encoding of Section 3.1, Lemma 5 and the confluence results for orthogonal CRSs (see for example [20,21]).

Similarly, we can deduce in the ρ-calculus interesting properties for a special kind of reductions, called developments. Intuitively, a development corresponds to the computation of a chosen set of reducible expressions in a term. 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. Interestingly, the notion of superdevelopments allows to derive a new second-order matching algorithm [11].

The general defintion of developments for the rewriting calculus can be found in [2]. 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!):

Corollary 2 (Finite developments).

- *Developments in the ρ-calculus are always finite.*
- *All developments of a ρ-term t end on the same final term.*

Proof. Follows from the results on developments proved for CRSs (see [15,20]).

These properties of developments are a key hypothesis to achieve a standardisation result. We know from the two theorems FD and FD! 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. Standardisation ensures that, for any derivation, the reduction steps can always be reordered to obtain a derivation in a canonical form, called standard:

Corollary 3 (Standardisation). *For any two ρ-terms t_1 and t_2 such that t_1 rewrites to t_2, there exists a standard derivation leading from t_1 to t_2.*

Proof. Follows from the results on standardisation proved for CRSs (see *e.g.* [18]).

4 Conclusions

The ρ-calculus is a powerful framework for specifying and reasoning about computation and deduction. It is therefore of main interest to study the relationship of the calculus with similar ones, in particular to understand its capabilities and properties.

We have presented in this paper the simulation of the rewriting calculus into *Combinatory Reduction systems*, the converse simulation having already been addressed in [3]. We have shown the encoding of the ρ-calculus into an appropriate CRS having as set of terms the encoding of ρ-terms and as set of rewrite rules the encoding of the evaluation rules of the ρ-calculus. We have then proved the soundness and completeness of this translation. These results allow us to adapt the well-developed CRSs meta-theory to the rewriting calculus. In particular, we can derive the properties of confluence, finite developments and standardisation for the ρ-calculus. Similar approaches can be applied to related calculi like the lambda calculus with patterns [19] or the pattern calculus [14].

The natural encoding we have presented in the paper leads to a simulation step-by-step of ρ-derivations into CRS-derivations. The translation the other way round was not so neat, since "walking through the context" is done implicitly in CRSs and thus additional ρ-terms needed to be inserted to direct the reduction in the ρ-calculus. Here instead, the explicit application operator of the ρ-calculus and the definition of rewrite rules at the object level in the ρ-calculus are encoded into CRS-terms using appropriate functional symbols. This allows to maintain, in the translated terms, the control on the position to which the rewrite rule is applied, which is a typical ingredient of the ρ-calculus.

We have considered in this paper the ρ-calculus with syntactic matching and the two evaluation rules (δ) and (ρ). The obtained results can be easily generalised to the version of the ρ-calculus with explicit delayed matching constraints. This version of the ρ-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 [6]. 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. This new symbol can be encoded in a CRS-term using additional functional symbols in the CRS-signature, similarly as for the other ρ-calculus operators. The encoding of other versions of the ρ-calculus into CRSs is matter of further study.

Other higher-order rewrite systems have already been compared. In particular, it has been shown that CRSs and HRSs have the same expressive power and therefore they can be considered equivalent [21]. Using this comparison, we can have an indirect representation of the ρ-calculus into HRSs that is based on the translation from the CRSs to HRSs and the translation from ρ-calculus to CRSs we have defined in this paper.

Acknowledgements: Many thanks to Vincent van Oostrom for fruitful suggestions on the most natural encoding of the ρ-calculus as a CRS and to the anonymous referees for their valuable remarks and suggestions.

References

1. G. Barthe, H. Cirstea, C. Kirchner, and L. Liquori. Pure Patterns Type Systems. In *Principles of Programming Languages - POPL'03*, pages 250–261. ACM, 2003.

2. C. Bertolissi. Developments in the rewriting calculus. Research Report http://hal.inria.fr/inria-00121212, INRIA & LORIA, 2006.
3. C. Bertolissi, H. Cirstea, and C. Kirchner. Expressing combinatory reduction systems derivations in the rewriting calculus. *Higher-Order and Symbolic Computation*, 19(4):345–376, dec 2006.
4. P. Brauner. Un calcul des séquents extensible. Master thesis, LORIA, 2006.
5. H. Cirstea. *Calcul de réécriture : fondements et applications.* Thèse de Doctorat d'Université, Université Henri Poincaré - Nancy I, 2000.
6. H. Cirstea, G. Faure, and C. Kirchner. A rho-calculus of explicit constraint application. In *Proc. of WRLA'2004*, volume 117 of *ENTCS*, 2004.
7. H. Cirstea and C. Kirchner. The rewriting calculus as a semantics of ELAN. In J. Hsiang and A. Ohori, editors, *4th Asian Computing Science Conference*, volume 1538 of *LNCS*, pages 8–10. Springer-Verlag, 1998.
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, 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, L. Liquori, and B. Wack. Rewrite strategies in the rewriting calculus. In B. Gramlich and S. Lucas, editors, *Proceedings of WRS'03*. ENTCS, June 2003.
11. G. Faure. Matching modulo superdeveloppements. application to second-order matching. In *Logic for Programming Artificial Intelligence and Reasoning*, Phnom Penh, 2006.
12. C. Houtmann. Cohérence de la déduction surnaturelle. Master thesis, LORIA, 2006.
13. G. Huet. *Résolution d'équations dans les langages d'ordre 1,2, ...,ω.* Thèse de Doctorat d'Etat, Université de Paris 7 (France), 1976.
14. B. Jay and D. Kesner. Pure pattern calculus. In P. Sestoft, editor, *15th European Symposium on Programming Languages and Systems*, volume 3924 of *Lecture Notes in Computer Science*, pages 100–114, Vienna (Austria), mar 2006. Springer Verlag.
15. J. W. Klop. *Combinatory Reduction Systems.* PhD thesis, CWI, 1980.
16. J. W. Klop, V. van Oostrom, and F. van Raamsdonk. Combinatory Reduction Systems: Introduction and Survey. *Theoretical Computer Science*, 121:279–308, 1993.
17. L. Liquori and B. Serpette. iRho: an Imperative Rewriting Calculus. In *Proc. of PPDP'04*, pages 167–178. The ACM Press, 2004.
18. P.-A. Melliès. *Description Abstraite des Systèmes de Réécriture.* PhD thesis, Université Paris 7, 1996.
19. V. van Oostrom. Lambda calculus with patterns. Technical report, Vrije Universiteit, Amsterdam, Nov. 1990.
20. V. van Oostrom. *Confluence for Abstract and Higher-Order Rewriting.* PhD thesis, Vrije Universiteit, Amsterdam, 1994.
21. V. van Oostrom and F. van Raamsdonk. Comparing combinatory reduction systems and higher-order rewrite systems. In *Higher-Order Algebra, Logic and Term Rewriting (HOA)*, pages 276–304, 1993.
22. B. Wack. *Typage et déduction dans le calcul de réécriture.* Thèse de doctorat, Université Henri Poincaré - Nancy I, Oct 2005.