Electronic Notes in Theoretical Computer Science 86 No. 2 (2003)
URL: http://www.elsevier.nl/locate/entcs/volume86.html 17 pages

Translating *Combinatory Reduction Systems* into the *Rewriting Calculus*

Clara Bertolissi, and Horatiu Cirstea, and Claude Kirchner

INPL & University Nancy II & INRIA & LORIA
615, rue du Jardin Botanique, BP-101
54602 Villers-lès-Nancy 54506 France
{Clara.Bertolissi,Horatiu.Cirstea,Claude.Kirchner}@loria.fr

Abstract

The last few years have seen the development of the *rewriting calculus* (or rho-calculus, ρCal) that extends first order term rewriting and λ-calculus. The integration 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*, or by adding to λ-calculus algebraic features. The different higher-order rewriting systems and the rewriting calculus share similar concepts and have similar applications, and thus, it seems natural to compare these formalisms. We analyze in this paper the relationship between the Rewriting Calculus and the Combinatory Reduction Systems and we present a translation of *CRS*-terms and rewrite rules into rho-terms and we show that for any *CRS*-reduction we have a corresponding rho-reduction.

1 Introduction

Lambda calculus and term rewriting provide two fundamental computational paradigms that had a deep influence on the development of programming and specification languages, and on proof environments. Starting from Klop's ground-breaking work on higher-order rewriting, and because of their complementarity, many frameworks have been designed with a view to integrate these two formalisms.

This integration has been handled either by enriching first-order rewriting with higher-order capabilities or by adding to λ-calculus algebraic features. In the first case, we find the works on CRS [15] and other higher-order rewriting systems [23,20], in the second case the works on combination of λ-calculus with term rewriting [1,5,12] to mention only a few.

For example, the *Combinatory Reduction Systems* (*CRS*), introduced by J.W.Klop [16] are an extension of first order rewrite systems with a mechanism of bound variables like in the λ-calculus. The meta-language of *CRS*, *i.e.* the

language in which the notions of substitution and rewrite step are expressed, is based on λ-calculus and higher-order matching (matching modulo the β-rule).

In the same line, the Rewriting Calculus (also called ρCal) [7,8,10], extends first order term rewriting and λ-calculus. Its main design concept is to make all the basic ingredients of rewriting explicit objects, in particular the notions of rule *application* and *result*. By making the rule application explicit, the calculus emphasises on one hand the fundamental role of matching and on the other hand the intrinsic higher-order nature of rewriting. The Rho Calculus offers a broad spectrum of applications due to the two fundamental parameters of the calculus: the theory modulo which matching is performed and the structure under which the results of a rule application are returned. Adjusting these parameters to various situations permits us to easily describe in a uniform but still appropriately tuned manner different calculi, like, for example, lambda calculus, term rewriting and object calculi.

Since the different higher-order rewriting systems and the rewriting calculus share similar concepts and have similar applications, it seems natural to compare these formalisms. The comparison between different higher-order formalisms, like for example between *Combinatory Reduction Systems* and *Higher-order Rewrite Systems*, has already been done [22].

This paper is concerned with the analysis of the relation between the rewriting calculus and higher order rewriting. In particular we study the representation of *Combinatory Reduction Systems* in the rewriting calculus.

This work of analysis and comparison between the ρCal and the *CRS* is meant to better understand the behavior of these systems, in particular how the rewrite rules are applied to terms and how term reductions are performed in the *CRS*. The main contribution of this paper is the definition of a translation of the various *CRS* concepts, like terms, assignments, matching etc. to the corresponding notions of the ρCal, and, using this translation, the proof that every reduction of a *CRS*-term can be reproduced in the ρCal.

The paper is structured as follows: In Section 2 we briefly present the ρCal through its components. In Section 3 we give a description of *CRS*s and we give some examples. In Section 4 we present a translation from *CRS*-terms into ρ-terms and we state a theorem on the correspondence between *CRS*-reductions and ρ-reductions. Section 5 concludes the paper and gives some perspectives to this work. The detailed proofs as well as some more examples can be found in [4].

2 The rewriting calculus

We briefly present in what follows the syntax and the semantics of the ρCal. For a more detailed presentation the reader can refer to [8,10] and for a typed version of the calculus to [3,9].

$$
\begin{array}{lll}
(\rho)\ (t_1 \rightarrowtriangle t_2)t_3 & \rightarrow_\rho & [t_1 \ll t_3].t_2 \\
(\sigma)\ [t_1 \ll t_3].t_2 & \rightarrow_\sigma & \sigma_1(t_2), \ldots, \sigma_n(t_2), \ldots \\
 & & \textit{where } \sigma_i \in \mathit{Sol}(t_1 \lessdot_{\mathbb{T}} t_3) \\
(\delta)\quad (t_1, t_2)\, t_3 & \rightarrow_\delta & t_1\, t_3, t_2\, t_3
\end{array}
$$

Fig. 1. Small-step reduction semantics

2.1 Syntax

In this paper, the symbols $t, u, \ldots$ range over the set $\mathcal{T}$ of terms, the symbols $X, Y, Z, \ldots, x, y, z \ldots$ range over the infinite set $\mathcal{X}$ of variables (we usually use capitals for free variables) and the symbols $f, g, \ldots$ range over the infinite set $\mathcal{F}$ of constants. All symbols can be indexed. The set of ρ-terms is defined as follows:

$$\mathcal{T} ::= \mathcal{X} \mid \mathcal{F} \mid \mathcal{T} \rightarrowtriangle \mathcal{T} \mid [\mathcal{T} \ll \mathcal{T}].\mathcal{T} \mid \mathcal{T}\ \mathcal{T} \mid \mathcal{T}, \mathcal{T}$$

We assume that the (hidden) application operator to the left, while the other operators associate to the right. The priority of the application operator is higher than that of "$[\ll].$" which is higher than that of the "$\rightarrowtriangle$" which is, in turn, of higher priority than the ",". By abuse of notation, function application is denoted by $t_0(t_1 \cdots t_n)$ instead of $t_0\ t_1 \cdots t_n$.

To support the intuition, we mention here that the application of an abstraction $t_1 \rightarrowtriangle t_3$ to a term t_2 always "fires" and produces as result the term $[t_1 \ll t_2].t_3$ which represents a constrained term where the matching equation is "put on the stack". The body of the constrained term will be evaluated or delayed according to the result of the corresponding matching problem. If a solution exists, the delayed matching constraint self-evaluates to $\sigma(t_3)$, where σ is the solution of the matching between t_1 and t_2. Finally, terms can be grouped together into *structures* built using the symbol ",".

As in any calculus involving binders, we work modulo the "*α-convention*" of Church [6], and modulo the "*hygiene-convention*" of Barendregt [2].

We should mention that there are two operators binding variables: in $t_1 \rightarrowtriangle t_2$, the free variables of t_1 are bound in t_2 and in $[t_1 \ll t_2].t_3$, the free variables of t_1 are bound in t_3 but not in t_2.

2.2 Semantics

The small-step reduction semantics is defined by the reduction rules presented in Figure 1. The central idea of the (ρ) rule of the calculus is that the application of a term $t_1 \rightarrowtriangle t_2$ to a term t_3, reduces to the delayed matching constraint $[t_1 \ll t_3].t_2$, while the application of the (σ) rule consists in solving (modulo the theory $\mathbb{T}$) the matching equation $t_1 \lessdot_{\mathbb{T}} t_3$, and applying the obtained result to the term t_2. The rule (δ) deals with the distributivity of

the application on the structures built with the "," constructor.

According to the matching theory specified [8], the application of the (σ) rule can produce an infinite number of substitutions as result. In the following we will restrict to matching theories leading to a finite (and even unitary) set of substitutions. The substitutions obtained as solution of a matching problem has the form $\sigma = \{x_1/t_1 \dots x_m/t_m\}$ where $Dom(\sigma) = \{x_1, \dots, x_m\}$. The application of a substitution σ to a term t, denoted by $\sigma(t)$ or $t\sigma$, can be straightforwardly adapted to deal with the new forms of constrained terms introduced in the ρCal(see [10]).

As usual, we introduce the classical notions of one-step, many-steps, and congruence relation of $\to_{\rho\sigma\delta}$. Let $\mathsf{Ctx}[-]$ be any context with a single hole, and let $\mathsf{Ctx}[t]$ be the result of filling the hole with the term t. The one-step evaluation $\mapsto_{\rho\sigma\delta}$ is defined by the following inference rules:

$$(\mathsf{Ctx}[-]) \quad \frac{t_1 \to_{\rho\sigma\delta} t_2}{\mathsf{Ctx}[t_1] \mapsto_{\rho\sigma\delta} \mathsf{Ctx}[t_2]} \qquad \forall\ t_1, t_2 \in \mathcal{T}$$

where $\to_{\rho\sigma\delta}$ denotes one of the top-level rules of ρCal. The many-step evaluation $\mapsto\!\!\!\!\to_{\rho\sigma\delta}$ is defined as the reflexive and transitive closure of $\mapsto_{\rho\sigma\delta}$. The congruence relation $=_{\rho\sigma\delta}$ is the symmetric closure of $\mapsto\!\!\!\!\to_{\rho\sigma\delta}$.

Example 2.1 [Small-step reductions] We take the terms

$$(f(\mathcal{X}) \twoheadrightarrow (3 \twoheadrightarrow 3)\mathcal{X})\, f(3) \quad \text{and} \quad (f(\mathcal{X}) \twoheadrightarrow (3 \twoheadrightarrow 3)\mathcal{X})\, f(4)$$

and we show one possible reduction for each of them (corresponding redexes are underlined and only the used rule (*e.g.* $\mapsto_\rho$) is shown instead of $\mapsto_{\rho\sigma\delta}$):

(i) $\underline{(f(\mathcal{X}) \twoheadrightarrow (3 \twoheadrightarrow 3)\mathcal{X})\, f(3)} \mapsto_\rho \underline{(f(\mathcal{X}) \twoheadrightarrow [3 \ll \mathcal{X}].3)\, f(3)} \mapsto_\rho$
$\underline{[f(\mathcal{X}) \ll f(3)].([3 \ll \mathcal{X}].3)} \mapsto_\sigma \underline{[3 \ll 3].3} \mapsto_\sigma 3$

(ii) $\underline{(f(\mathcal{X}) \twoheadrightarrow (3 \twoheadrightarrow 3)\mathcal{X})\, f(4)} \mapsto_\rho \underline{[f(\mathcal{X}) \ll f(4)].((3 \twoheadrightarrow 3)\, \mathcal{X})} \mapsto_\sigma$
$\underline{(3 \twoheadrightarrow 3)\, 4} \mapsto_\rho [3 \ll 4].3$

2.3 Two versions of the rewriting calculus

The general ρCal can be instantiated to simpler versions when the syntax is restricted and the matching theory used in the (σ) rule is specified [7]. For example, when all the ρ-rules are of the form $\mathcal{X} \twoheadrightarrow \mathcal{T}$ and a syntactic matching is used, a version similar to the lambda-calculus, denoted ρCal_λ, is obtained. We denote the reductions in the obtained calculus by $\to_{\rho_\lambda}$. Starting from the corresponding congruence relation $=_{\rho_\lambda}$, we can define the matching theory $\mathbb{T}_\lambda$ such that $\mathbb{T}_\lambda \models t_1 = t_2$ *iff* $t_1 =_{\rho_\lambda} t_2$.

More powerful versions of the ρCal can be obtained using more elaborated (decidable) matching theories like, for example, pattern matching [17]. Our goal is to define a version of the ρCal with an evaluation mechanism similar to that of CRSs and thus using a (pattern) higher-order matching. Therefore, we introduce now the notion of ρ-pattern directly inspired from the *CRS*-patterns (defined in Section 3).

Definition 2.2 [ρ-pattern] A ρ-term p is called a *ρ-pattern* if any of its free variables Z appears in a sub-term of p of the form $Z\ x_1 \dots\ x_n$ where the variables $x_1, \dots, x_n$, $n \geq 0$, are distinct and all bound in p.

For example, $x \twoheadrightarrow f(Z\ x)$ is a ρ-pattern while $x \twoheadrightarrow (Z\ x\ x)$ and $g(x \twoheadrightarrow x, Z\ x)$ are not.

The $\rho Cal_{\P}$ is then obtained as the ρCal whose rules are of the form $\mathcal{P} \twoheadrightarrow \mathcal{T}$, with $\mathcal{P}$ the set of ρ-patterns, and using a matching modulo $\mathbb{T}_\lambda$ (denoted $\lessdot_{\P}$). Since pattern matching is decidable and unitary [17] the matching involved in the application of the evaluation rule (σ) of the $\rho Cal_{\P}$ yields a single substitution.

3 Combinatory Reduction Systems

The *Combinatory Reduction Systems* (*CRS*), introduced by J.W.Klop in 1980 [16], are a generalization of first-order term rewrite systems with a mechanism of bound variables like in the λ-calculus. For defining the components of a *CRS* we refer to [15].

3.1 Syntax

In what follows the symbols $A, B, \dots L, P, R, \dots$ range over the set $\mathcal{MT}$ of metaterms, the symbols $t, u, \dots$ range over the set $\mathcal{T}_{\mathcal{CRS}}$ of terms, the symbols $x, y, z, \dots$ range over the set $\mathcal{X}$ of variables, the symbols $X, Y, Z \dots$ range over the set $\mathcal{Z}$ of metavariables of fixed arity and the (functional) symbols $f, g, \dots$ range over the set $\mathcal{F}$ of symbols of fixed arity . All symbols can be indexed. We denote by $\equiv$ the syntactic identity of metaterms or substitutions.

The set of *CRS*-metaterms $\mathcal{MT}$, is defined as follows:

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

The set $\mathcal{T}_{\mathcal{CRS}} \subset \mathcal{MT}$ of *CRS*-terms is composed of all the metaterms without metavariables.

Comparing to first-order rewrite systems, in the syntax of a *CRS* we have two new concepts: the symbol $[\_]\_$ and the metavariables. The operator $[\_]\_$ denotes an abstraction similar to the λ-abstraction of the λ-calculus such that, in $[x]t$, the variable x is bound in t. The variables bound by $[\_]\_$ may be renamed by α-conversion. Metavariables (in *CRS* rewrite rules) behave as free variables of first-order rewrite systems. Metavariables cannot be bound by the abstraction operator, but can depend on bound variables by means of their arguments. The set of metavariables of a metaterm A is written $\mathcal{MV}(A)$. A metaterm is called *closed* if all its variables occur bound.

A *CRS* rewrite rule is a couple of metaterms. Their metavariables define the reduction schemes since they can be instantiated with the value

of all possible terms. We consider as left-hand side of the rules only the *CRS*-metaterms satisfying the pattern definition:

Definition 3.1 [*CRS*-pattern] A *CRS*-metaterm P is said to be 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.

Moreover, the usual conditions used in first-order rewriting are imposed and thus, for a *CRS* rewrite rule $L \to R$ we have: L and R are closed, L has the form $f(A_1, \ldots, A_n)$ with $A_1, \ldots, A_n$ metaterms and $f \in \mathcal{F}$ of arity n, $\mathcal{MV}(L) \supseteq \mathcal{MV}(R)$ and L is a *CRS* pattern. The last restriction ensures the decidability and the uniqueness of the solution of the matching inherent to the application of the *CRS*-rules.

Example 3.2 [β-rule in *CRS*] The β-rule of λ-calculus $(\lambda x.t)u \longrightarrow_\beta t\{x/u\}$ corresponds in *CRS* to the rewrite rule $BetaCRS$:

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

where $App \in \mathcal{F}$ of arity 2 and $Ab \in \mathcal{F}$ of arity 1 are the encodings for the application operator and the abstraction operator respectively.

3.2 Semantics

Given a rewrite rule $L \to R$ and a substitution σ (also called assignment, as defined below), we have $\sigma(L) \mapsto_{L \to R} \sigma(R)$ if $\sigma L, \sigma R \in \mathcal{T}_{\mathcal{CRS}}$. 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 $\mapsto_{\mathcal{R}}$ (denoted also $\mapsto_{L \to R}$ if we want to specify the applied rule) is the context closure of the 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 $\mapsto_{\mathcal{R}}$.

The application of substitutions to metavariables is defined at the meta-level of the calculus and uses $\underline{\lambda}$-calculus as meta-language (just for distinguishing it from classical "lambda"). Unintended bindings of variables by the $\underline{\lambda}$-abstractor operator are avoided using α-conversion. The reduction of $\underline{\lambda}$-redexes is performed by the $\underline{\beta}$-rule of the $\underline{\lambda}$-calculus. We denote by $t{\downarrow_{\underline{\beta}}}$ the $\underline{\beta}$-normal form of the term t. We should point out that a *CRS*-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*-metaterm.

An n-ary *substitute* [14] 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 ad its application to an n-tuple of *CRS*-terms $(t_1, \ldots, t_n)$ yields the simultaneous substitution $(\underline{\lambda} x_1 \ldots x_n.u)(t_1, \ldots, t_n){\downarrow_{\underline{\beta}}} = u\{x_1 := t_1, \ldots, x_n := t_n\}$

Definition 3.3 [Assignment] An *assignment* $\sigma = \{(Z_1, \xi_1), \ldots, (Z_n, \xi_n)\}$, is a finite set of pairs (metavariable, substitute) such that $\text{arity}(Z_i) = \text{arity}(\xi_i)$ $\forall i \in \{1, \ldots, n\}$. The application of an assignment σ to a *CRS*-metaterm t, denoted $\sigma(t)$ or $t\sigma$, is inductively defined in the following way:

$$\sigma([x]t) \triangleq [x]\sigma(t) \qquad \sigma(f(t_1, \ldots, t_n) \triangleq f(\sigma(t_1), \ldots, \sigma(t_n))$$

$$\sigma(Z_i) \triangleq \xi_i \quad \text{if } (Z_i, \xi_i) \in \sigma \qquad \sigma(Z_i(t_1, \ldots, t_n)) \triangleq \sigma(Z_i)(\sigma(t_1), \ldots, \sigma(t_n))\downarrow_{\underline{\beta}}$$

Therefore the instantiation of rewrite rules in order to obtain an actual rewrite step is defined by replacing each metavariable by a $\underline{\lambda}$-term and by reducing all residuals of $\underline{\beta}$-redexes that are present in the initial term, i.e. performing a development on $\underline{\lambda}$-terms. Since in λ-calculus all developments are finite, the *CRS* substitution is well-defined. Note that the result of the application of an assignment to a metaterm is indeed a term.

To determine the assignment that applied to a metaterm leads to a given term, the concept of matching is used. *CRS* uses higher-order matching but the way the assignment is obtained has not been clearly specified until now.

Example 3.4 Given the *CRS*-term $f(t)$ with $t = App(Ab([x]f(x)), a)$. We apply to the sub-term t the *BetaCRS* rule (Example 3.2). A solution of the corresponding matching problem is the assignment $\sigma = \{(Z, \underline{\lambda}y.fy), (Z_1, a)\}$ since when applying it to the the left-hand side L of the rule *BetaCRS*, we have $\sigma(L) = \sigma(App(Ab([x]Z(x)), Z_1)) = App(Ab([x]\sigma(Z)(x), \sigma(Z_1)) = App(Ab([x](\underline{\lambda}y.fy)(x), a))\downarrow_{\underline{\beta}} = App(Ab([x]f(x)), a) = t$.

We obtain $\sigma(R) = \sigma(Z(Z_1)) = (\sigma(Z))(\sigma(Z_1)) = (\underline{\lambda}y.fy)(a) \downarrow_{\underline{\beta}} = f(a)$ where R is the right-hand side of the rule *BetaCRS*. Therefore we have $t \mapsto_{L \to R} f(a)$ and thus $f(t) \mapsto_{L \to R} f(f(a))$.

4 Translating the *CRS* into the Rewriting Calculus

We propose in this section a translation of *CRS*-(meta)terms into ρ-terms and we show that to *CRS*-reductions correspond ρ-reductions. The assignment application used for performing term reductions in a *CRS* (and thus the matching the *CRS*-reduction relies on) is based on λ-calculus. Consequently, to encode all the expressiveness of *CRS* into the rewriting calculus, we need a greater matching power than the syntactic matching and for this reason we use the $\rho Cal_{\P}$ as target calculus.

In the following we suppose that the set of constants of the considered $\rho Cal_{\P}$ contains the set of functional symbols of the corresponding *CRS* and the set of variables of $\rho Cal_{\P}$ contains the variables and metavariables of the corresponding *CRS*.

Definition 4.1 [Translation] The translation of a *CRS*-metaterm t into a ρ-term, denoted $\overline{t}$ or $< t >$, is inductively defined as follows:

- *CRS*-terms into ρ-terms

$$\overline{x} \triangleq x \qquad \overline{f(t_1,\ldots,t_n)} \triangleq f(\overline{t_1}\ldots\overline{t_n})$$
$$\overline{[x]t} \triangleq x \twoheadrightarrow \overline{t} \qquad \overline{Z(t_1,\ldots,t_n)} \triangleq Z\ \overline{t_1}\ldots\overline{t_n}$$

- *CRS* rewrite rules into ρ-terms:

$$\overline{L \to R} \triangleq \overline{L} \twoheadrightarrow \overline{R}$$

- *CRS* substitutes into ρ-terms:

$$\overline{\underline{\lambda}x_1\ldots x_n.u} \triangleq x_1 \twoheadrightarrow (x_2 \twoheadrightarrow (\ldots(x_n \twoheadrightarrow \overline{u})\ldots))$$

- *CRS* assignments into ρ-substitutions:

$$\overline{\{\ldots,(Z,\underline{\lambda}x_1\ldots x_n.u),\ldots\}} \triangleq \{\ldots,Z/\ x_1 \twoheadrightarrow (x_2 \twoheadrightarrow (\ldots(x_n \twoheadrightarrow \overline{u})\ldots)),\ldots\}$$

We can observe that the translation of the *CRS*-abstraction operator "[ ]" corresponds to the ρ-abstraction operator "$\twoheadrightarrow$". An n-ary *CRS*-metavariable (function) corresponds in the $\rho Cal_{\P}$ to a variable (constant) applied to n ρ-terms.

Since in ρCal rewrite rules are first class objects, a *CRS* rewrite rule is translated into a ρ-term and more precisely into a ρ-rule.

The $\underline{\lambda}$-abstraction operator defined at the meta-level of *CRS* is translated into the ρ-abstraction operator "$\twoheadrightarrow$". This means that reductions performed in *CRS* at the meta-level (using $\underline{\lambda}$) correspond in the $\rho Cal_{\P}$ to explicit reductions corresponding to the application of the abstraction operator "$\twoheadrightarrow$".

The translation of the abstraction operator and of the rewrite rules of a *CRS* into the same abstraction operator of the ρCal corresponds to the uniform treatment of first and higher-order rewriting in the ρCal.

Example 4.2 We have already seen how the β-rule of λ-calculus can be translated into a *CRS*. When translating this *BetaCRS* rule into the rewriting calculus the following term is obtained:

$$\overline{BetaCRS} \triangleq App(Ab(x \twoheadrightarrow (Z\ x)), Z_1) \twoheadrightarrow Z\ Z_1$$

where $App, Ab \in \mathcal{F}$ and $x, Z, Z_1 \in \mathcal{X}$.

The *CRS*-abstraction operator is never directly applied to a *CRS*-term, since we have no application symbols in the syntax of *CRS*. Nevertheless it is translated into the ρ-abstraction operator ensuring that the corresponding variables are bound in the translation and thus, the preservation of the pattern condition by the translation.

Lemma 4.3 *Given a* CRS*-metaterm L. If L satisfies the* CRS *pattern definition, then $\overline{L}$ satisfies the ρ-pattern definition.*

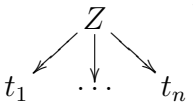
Proof. Follows immediately from Definition 4.1 and Definition 2.2. □

Moreover, as a consequence of the definition of the translation, we have that every ρ-term obtained from a *CRS*-metaterm does not contain any redexes and thus, it is in normal form.

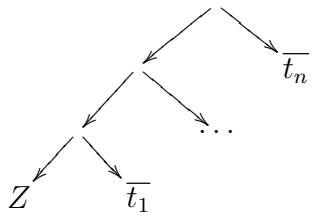
We show in the following that we can express *CRS*-derivations in the rewriting calculus using the translation we have proposed above. We state some lemmas and the main theorems and we give an example of *CRS*-reduction and its corresponding one in the $\rho Cal_{\P}$.

First of all we need some notation facilities. For any metaterm A we denote by $\mathcal{P}os(A)$ the set of all possible positions in A. We call ϵ the head position of A. $A(\omega)$ is the symbol at the position ω in A. A sub-metaterm of A at the position $\omega \in \mathcal{P}os(A)$ is denoted $A_{|\omega}$ and defined by $\forall \omega.\omega' \in \mathcal{P}os(A), \omega' \in \mathcal{P}os(A_{|\omega}), A_{|\omega}(\omega') = t(\omega.\omega')$. We use the notation $A_{\lceil B \rceil_\omega}$ to signify that A has a sub-metaterm B at the position ω. To simplify the notation we write $A(\omega.i^n.\omega')$ for $A(\omega.\underbrace{(i \ldots i)}_{n}.\omega')$.

The position ω of a sub-metaterm B in a *CRS*-metaterm $A_{\lceil B \rceil_\omega}$ and the position of its translation $\overline{B}$ in the ρ-term $\overline{A}$ are not the same. This is due to the different use of (meta)variables and functional symbols in the *CRS*- and ρ-(meta)terms. Indeed, since an n-ary metavariable and an n-ary function are translated in the $\rho Cal_{\P}$ as a variable or function applied to n ρ-terms, the n-ary tree of the *CRS*-metaterm



becomes a binary tree in ρCal



The following function $tr(\_,\_)$ defines the changing of position after the translation.

Definition 4.4 [Position transformation] Let A be a *CRS*-metaterm and $\omega \in \mathcal{P}os(A)$ a position in the metaterm A. The transformation function $tr(\omega, A) : \mathcal{P}os(A) \times \mathcal{MT} \mapsto \mathcal{P}os(\overline{A})$ is inductively defined in the following way:

- $tr(\epsilon, A) = \epsilon$
- $tr(n - i.\omega, Z(A_1, \ldots, A_n)) = 1^i.2.tr(\omega, A_{n-i})$ where $i \in \{0, \ldots, n-1\}$
- $tr(n - i.\omega, f(A_1, \ldots, A_n)) = 1^i.2.tr(\omega, A_{n-i})$ where $i \in \{0, \ldots, n-1\}$
- $tr(i.\omega, [x]A) = 1.\epsilon$ if $i = 1$ and $\omega = \epsilon$
  $= i.tr(\omega, A)$ if $i = 2$
  $= wrong$ in all other cases

The correctness of this function w.r.t. the translation can be shown by structural induction on the *CRS*-metaterm:

Lemma 4.5 *Let $A_{\lceil B \rceil_\omega}$ be a* CRS*-metaterm with a sub-term B at the position ω. Then:*

$$\overline{A_{\lceil B \rceil_\omega}} = \overline{A}_{\lceil \overline{B} \rceil_{tr(\omega,A)}}$$

We show that the translation "preserves" the matching solution, *i.e.* for every assignment σ permitting the application of a *CRS* rewrite rule to a *CRS*-term, the translation of the rule into the $\rho Cal_{\P}$ can be applied to the translation of the corresponding term using the corresponding substitution.

Lemma 4.6 *Let A be a* CRS*-metaterm and σ an assignment. Then*

$$\overline{\sigma}(\overline{A}) \ \mapsto\!\!\!\!\!\rightarrow_{\rho_\lambda} \ \overline{\sigma(A)}$$

Proof. By structural induction on the term A. □

As an immediate consequence we obtain as well $\overline{\sigma}(\overline{A}) \ \mapsto\!\!\!\!\!\rightarrow_{\rho\sigma\delta} \ \overline{\sigma(A)}$ and $\overline{\sigma}(\overline{A}) \ =_{\rho_\lambda} \ \overline{\sigma(A)}$. This result is important since $\mathbb{T}_\lambda$ is the matching theory of the $\rho Cal_{\P}$ and thus, the $\mapsto\!\!\!\!\!\rightarrow_{\rho_\lambda}$ reductions are considered when testing whether a substitution is the solution of a given matching problem.

The $\mathbb{T}_\lambda$ matching theory is effectively used when the *CRS*-metaterm A contains metavariables of arity different from zero. In that case the application of the assignment involves $\underline{\beta}$-reduction steps performed at the meta-level of the *CRS*-reduction and these steps correspond to explicit ρ-reductions.

Using the above lemmas we can show that starting from a *CRS*-reduction a corresponding reduction in $\rho Cal_{\P}$ can be obtained. For this, a ρ-term encoding the *CRS*-derivation trace is constructed. When only a one-step *CRS*-reduction is considered, *i.e.* only one *CRS*-rule is applied, the corresponding ρ-term depends on the initial term to be reduced and on the applied rewrite rule.

Furthermore, we show that every possible ρ-derivation resulting from the correct initial ρ-term terminates and converges to the correct result.

Theorem 4.7 *Let t_0, t_1 be two* CRS*-terms, $L \to R$ a* CRS*-rule and σ an assignment such that $t_{0\lceil \sigma(L) \rceil_\omega}$ and $t_1 \equiv t_{0\lceil \sigma(R) \rceil_\omega}$. Given the ρ-term $u_0 \triangleq \overline{t_0}_{\lceil x \rceil_{tr(\omega,t_0)}} \twoheadrightarrow \overline{t_0}_{\lceil \overline{L \to R} \ x \rceil_{tr(\omega,t_0)}}$, then, every derivation resulting from $u_0 \ \overline{t_0}$ terminates and converges to $\overline{t_1}$:*

$$(u_0 \ \overline{t_0}) \ \mapsto\!\!\!\!\!\rightarrow_{\rho\sigma\delta_{\P}} \ \overline{t_1}$$

Proof. Thanks to the form of the ρ-term u_0 that permits to apply the rewrite rule exactly at the needed position, the ρ-reduction follows relatively easily. For the sake of readability we only show the case $\omega = \epsilon$.

$(u_0 \ \overline{t_0}) \triangleq (x \twoheadrightarrow ((\overline{L \to R}) \ x)) \ \overline{t_0} \triangleq (x \twoheadrightarrow ((\overline{L \to R}) \ x)) \ \overline{\sigma(L)}$

$\triangleq (x \twoheadrightarrow ((\overline{L} \twoheadrightarrow \overline{R}) \ x)) \ \overline{\sigma(L)}$ (By Definition 4.1)

$\mapsto_\rho [x \ll \overline{\sigma(L)}].((\overline{L} \twoheadrightarrow \overline{R})\ x)$

$\mapsto_\sigma (\overline{L} \twoheadrightarrow \overline{R})\ \overline{\sigma(L)}$

$\mapsto_\rho [\overline{L} \ll \overline{\sigma(L)}].\overline{R}$

$= [\overline{L} \ll \overline{\sigma}(\overline{L})].\overline{R}$ (By Lemma 4.6)

$\mapsto_\sigma \overline{\sigma}(\overline{R})$

$\mapsto\!\!\!\!\to_{\rho\sigma\delta} \overline{\sigma(R)}$ (By Lemma 4.6)

$\triangleq \overline{t_1}$

We use a little abuse of notation when stating that $[\overline{L} \ll \overline{\sigma(L)}].\overline{R} = [\overline{L} \ll \overline{\sigma}(\overline{L})].\overline{R}$ due to the fact that, by Lemma 4.6, the two corresponding matching problems $\overline{L} \prec\!\!\!\prec_{\P} \overline{\sigma(L)}$ and $\overline{L} \prec\!\!\!\prec_{\P} \overline{\sigma}(\overline{L})$ have the same solution. The reduction for $\omega \neq \epsilon$ is similar, the only difference being at the matching level. In this case we also use Lemma 4.5 when matching against the translation of the left-hand side of the rule.

Another possible reduction is the following one:

$(u_0\ \overline{t_0}) \triangleq (x \twoheadrightarrow ((\overline{L \to R})\ x))\ \overline{t_0} \triangleq (x \twoheadrightarrow ((\overline{L \to R})\ x))\ \overline{\sigma(L)}$

$\triangleq (x \twoheadrightarrow ((\overline{L} \twoheadrightarrow \overline{R})\ x))\ \overline{\sigma(L)}$ (By Definition 4.1)

$\mapsto_\rho (x \twoheadrightarrow [\overline{L} \ll x].\overline{R})\ \overline{\sigma(L)}$

$\mapsto_\rho [x \ll \overline{\sigma(L)}].([\overline{L} \ll x].\overline{R})$

$\mapsto_\sigma [\overline{L} \ll \overline{\sigma(L)}].\overline{R}$

$= [\overline{L} \ll \overline{\sigma}(\overline{L})].\overline{R}$ (By Lemma 4.6)

$\mapsto_\sigma \overline{\sigma}(\overline{R})$

$\mapsto\!\!\!\!\to_{\rho\sigma\delta} \overline{\sigma(R)}$ (By Lemma 4.6)

$\triangleq \overline{t_1}$

These are the only possible derivations since the translations of *CRS*-terms contain no redexes and the matching problem $\overline{L} \prec\!\!\!\prec_{\P} x$ in the latter derivation has no solution since L cannot be a variable. □

We can notice that we have a longer derivation scheme in the $\rho Cal_{\P}$ than in the *CRS*. For every rewrite step in the *CRS*, in the $\rho Cal_{\P}$ we have two (ρ)-rule steps plus two (σ)-rule steps for the application of the rewrite rule and some additional steps corresponding to the β-reduction steps performed at the meta-level of the *CRS*-reduction. These latter steps are performed at the object-level of the $\rho Cal_{\P}$ and their number depends on the arity of the *CRS*-metavariables in the right-side of the considered rewrite rule. We should point out that the matching performed at the meta-level of the $\rho Cal_{\P}$ may involve some derivations but this time performed in ρCal_{λ}.

We can generalize the theorem above and built, using the derivations of a

term t_0 in a *CRS*, a ρ-term with a reduction similar to the one of t_0 in the *CRS*.

Lemma 4.8 *Let t_0, t_n be two* CRS*-terms such that $t_0 \mapsto\!\!\!\!\!\rightarrow_{\mathcal{R}} t_n$ and $u_0, \ldots, u_n$ the corresponding derivation trace terms in the ρCal. Then every derivation resulting from $(u_n \ \ldots \ (u_0 \ \overline{t_0}))$ terminates and converges to $\overline{t_n}$.*

Proof. By induction on the number of derivation trace terms $u_0, \ldots, u_n$. □

We can state thus that we have a complete and correct translation of *CRS*-reductions to ρ-reductions. Given the simplicity of the translation the properties of the rewrite system, like for example the orthogonality, are preserved. As far as it concerns the corresponding *CRS*-reductions, properties like the termination and the confluence are also preserved due to the direct correspondence with the ρ-reductions.

The main difference between the two systems lays in the fact that rewrite rules and consequently their control (application position) are defined at the object-level of the ρCal while in the *CRS* the reduction strategy is left implicit. The possibility to control the application of rewrite rules is particularly useful when the rewrite system is not confluent or terminant. Moreover, while in the *CRS* the β-reduction is implicitly included in the application of the assignment, in the $\rho Cal_{\P}$ the corresponding reductions are performed explicitly.

The ρ-terms u_i can be built automatically starting from the *CRS*-reduction steps as stated in Theorem 4.7. It is obviously interesting to give a method for constructing this terms without knowing *a priori* the derivation from t_0 to t_n but only the set of rewrite rules to be applied. This needs the definition of iteration strategies and of strategies for the generic traversal of terms. This has been done for the initial version of the ρCal either by enriching the calculus with a new operator [7] or by adding an "exception handling mechanism" to the calculus [11]. We conjecture that these approaches that have already been used for encoding first order rewriting can be used for the *CRS* translation as well.

More recently, we have been working on a typed version of the ρCal which allows the definition of iterators and, as a consequence, allows one to represent reductions in first order rewriting. The use of this method for representing *CRS* reductions will be the object of a later study.

Example 4.9 Given the *CRS*-reduction $f(App(Ab([x]f(x)), a)) \mapsto\!\!\!\!\!\rightarrow_{\mathcal{R}} f(f(a))$ presented in Example 3.4. In order to obtain a similar reduction in the $\rho Cal_{\P}$, we consider the ρ-term $f(y) \twoheadrightarrow f(\overline{BetaCRS}\ y)$ (with $BetaCRS$ defined in Example 3.2) and we apply it to the translation of the initial *CRS*-term:

$$(f(y) \twoheadrightarrow f(\overline{BetaCRS}\ y))\ (\overline{f(App(Ab([x]f(x)), a))})$$
$$\mapsto_{\rho}\ [f(y) \ll f(App(Ab(x \twoheadrightarrow f(x)), a))].f(\overline{BetaCRS}\ y)$$
$$\mapsto_{\sigma}\ f(\overline{BetaCRS}\ App(Ab(x \twoheadrightarrow f(x)), a))$$

$$\equiv f((App(Ab(x \rightarrowtriangle (Z\ x)), Z_1) \rightarrowtriangle Z\ Z_1)\ App(Ab(x \rightarrowtriangle f(x)), a))$$
$$\mapsto_\rho\ f([App(Ab(x \rightarrowtriangle (Z\ x)), Z_1) \ll App(Ab(x \rightarrowtriangle f(x)), a)].Z\ Z_1)$$
$$\mapsto_\sigma\ f(Z\{Z/z \rightarrowtriangle f(z)\}\ Z_1\{Z_1/a\})$$
$$= f((z \rightarrowtriangle f(z))\ a)$$
$$\mapsto_\rho f([z \ll a].f(z))$$
$$\mapsto_\sigma f(f(a))$$

One can notice in Example 4.9 that the reductions in *CRS* and in rewriting calculus lead to the same final term, modulo the translation, but we do not have an one-to-one correspondence between the rewrite steps (the steps from $f((z \rightarrowtriangle f(z))\ a)$ to $f(f(a))$ are explicit only in the ρCal). The same behavior is obtained in the following (more complicated) example.

Example 4.10 [λ-calculus with surjective pairing]

Given a *CRS* with the following set of rewrite rules $\mathcal{R}$:

$$\begin{aligned}\mathcal{R} = \{&P_0 : \Pi_0(\Pi(X_1, X_2)) \to X_1,\\ &P_1 : \Pi_1(\Pi(X_1, X_2)) \to X_2,\\ &P : \Pi(\Pi_0 X_1, \Pi_1 X_1) \to X_1,\\ &BetaCRS\}\end{aligned}$$

where $X_1, X_2 \in \mathcal{Z}_0$, $\Pi \in \mathcal{F}_2$ (pair function), $\Pi_0, \Pi_1 \in \mathcal{F}_1$ (projections) and $BetaCRS$ as in Example 3.2.

We consider the *CRS*-term $t = App(Ab([z]\Pi(\Pi_1 z, \Pi_0 z)), \Pi(x_1, x_2))$ consisting in the application of the function $Ab([z]\Pi(\Pi_1 z, \Pi_0 z))$, that swaps the elements of a pair, to the pair $\Pi(x_1, x_2)$.

To reduce the *CRS*-term t we first apply the rule *BetaCRS* with the assignment $\sigma = \{(Z, \underline{\lambda} z.\Pi(\Pi_1 z, \Pi_0 z), (Z_1, \Pi(x_1, x_2))\}$ and we obtain:

$$\begin{aligned}\sigma(Z(Z_1)) &= (\sigma(Z))(\sigma(Z_1))\\ &= (\underline{\lambda} z.\Pi(\Pi_1 z, \Pi_0 z))(\Pi(x_1, x_2)) \downarrow_{\overline{\beta}}\\ &= \Pi(\Pi_1(\Pi(x_1, x_2)), \Pi_0(\Pi(x_1, x_2)))\end{aligned}$$

Next we apply the rules P_1 and P_0 to the first and second argument of Π respectively, using the assignment $\sigma' = \{(X_1, x_1), (X_2, x_2)\}$ and we obtain the final result:

$$\Pi(\Pi_1(\Pi(x_1, x_2)), \Pi_0(\Pi(x_1, x_2))) \mapsto_{P_1} \Pi(x_2, \Pi_0(\Pi(x_1, x_2))) \mapsto_{P_0} \Pi(x_2, x_1)$$

We translate now the example into the $\rho Cal_{\P}$. We translate the set of *CRS*

rewrite rules

$$\begin{aligned}
\overline{P_0} &\triangleq \Pi_0(\Pi(X_1, X_2)) \twoheadrightarrow X_1 \\
\overline{P_1} &\triangleq \Pi_1(\Pi(X_1, X_2)) \twoheadrightarrow X_2 \\
\overline{P} &\triangleq \Pi(\Pi_0 X_1, \Pi_1 X_1) \twoheadrightarrow X_1 \\
\overline{BetaCRS} &\triangleq App(Ab(x \twoheadrightarrow (Z\ x)))\ Z_1) \twoheadrightarrow Z\ Z_1
\end{aligned}$$

and the *CRS*-term t:

$$\overline{t} \triangleq App(Ab(z \twoheadrightarrow (\Pi(\Pi_1 z, \Pi_0 z))), \Pi(x_1, x_2))$$

Starting from the *CRS* reduction above, we build the following ρ-terms corresponding to the application of the $\overline{BetaCRS}$ and $\overline{P_0}, \overline{P_1}$ rules respectively

$$u_1 = \overline{BetaCRS}\ and\ u_2 = (\Pi(x, y) \twoheadrightarrow \Pi(\overline{P_1}\ x, \overline{P_0}\ y))$$

and we build the ρ-term: $u_2\ (u_1\ \overline{t})$. In a more automatic approach we should have built three terms corresponding to the three *CRS* reduction steps.

First of all, we perform the $\overline{BetaCRS}$ reduction step using the substitution $\overline{\sigma} = \{Z/\ z \twoheadrightarrow \Pi(\Pi_1 z, \Pi_0 z),\ Z_1/\ \Pi(x_1, x_2)\}$ and we obtain

$$\begin{aligned}
\overline{\sigma}(Z\ Z_1) &= \overline{\sigma}(Z)\ \overline{\sigma}(Z_1) \\
&= (z \twoheadrightarrow \Pi(\Pi_1 z, \Pi_0 z))\ \Pi(x_1, x_2) \\
&\mapsto_\rho [z \twoheadleftarrow \Pi(x_1, x_2)].\Pi(\Pi_1 z, \Pi_0 z) \\
&\mapsto_\sigma \Pi(\Pi_1(\Pi(x_1, x_2)), \Pi_0(\Pi(x_1, x_2)))
\end{aligned}$$

Next, we continue reducing the ρ-term obtained as intermediary result:

$$\begin{aligned}
&(\Pi(x, y) \twoheadrightarrow \Pi(\overline{P_1}\ x, \overline{P_0}\ y))\ \Pi(\Pi_1(\Pi(x_1, x_2)), \Pi_0(\Pi(x_1, x_2))) \\
&\mapsto_\rho [\Pi(x, y) \ll \Pi(\Pi_1(\Pi(x_1, x_2)), \Pi_0(\Pi(x_1, x_2)))].\Pi(\overline{P_1}\ x, \overline{P_1}\ y) \\
&\mapsto_\sigma \Pi(\overline{P_1}\ \Pi(\Pi_1(\Pi(x_1, x_2)), \overline{P_0}\ \Pi_0(\Pi(x_1, x_2)))
\end{aligned}$$

The last reduction consists in applying the ρ-rules $\overline{P_0}$ and $\overline{P_1}$ using the substitution $\overline{\sigma'} = \{(X_1, x_1), (X_2, x_2)\}$.

$$\Pi(\overline{P_1}\ \Pi(\Pi_1(\Pi(x_1, x_2)), \overline{P_0}\ \Pi_0(\Pi(x_1, x_2))) \mapsto\!\!\!\!\!\!\rightarrow_{\rho\sigma} \Pi(x_2, x_1)$$

5 Conclusions

The applications of the rewriting calculus are various and numerous. The rewriting calculus is a sufficiently powerful framework allowing one to represent the usual computational formalisms. It contains the complementary properties of first-order rewriting and lambda calculus. Moreover, it permits the description of rewrite based languages and of the object oriented calculi in a natural and simple way. We have shown in this paper that also higher order

rewriting can be represented in the ρCal and in particular we have analyzed the relation with the *Combinatory Reduction systems*. Any reduction of a term *w.r.t.* a given *CRS* can be represented by a corresponding ρ-term.

This ρ-term can be built automatically starting from the *CRS*-reduction steps. We conjecture that the different approaches used for the representation of first-order rewriting in the ρCal can be applied here for the construction of an appropriate term only from the set of *CRS*-rules and without any knowledge on the reduction steps. In fact, the use of a recent definition of iterators in ρCal for the representation of reduction strategies like innermost and in particular for the representation of *CRS*-reductions is a work in progress.

The results of the comparison indicates a certain gap between the two formalisms. "Walking through the context" is done implicitly in the *CRS*, while additional ρ-terms need to be inserted to direct the reduction in the ρCal. Rewrite rules are defined at the object level of the ρCal and they are applied explicitly. The reduction is then performed by the three evaluation rules of the ρCal. On the contrary, in the *CRS* we have a set of rewrite rules that is particular to the *CRS* considered and the strategy of application is left implicit. Moreover, the evaluation of an assignment is done at the meta-level of the *CRS* using meta λ-calculus, while in the ρCal the application of a substitution leads to additional explicit reduction steps. For this reason, we generally have a longer reduction scheme in the ρCal than in the *CRS*.

Since we are mainly interested in the expressive power of the ρCal we have proposed in this paper a translation from *CRS* to ρCal but the translation the other way round has not been explicitly defined here. We believe this translation is possible but maybe not as obvious as one may think since the explicit control of rewrite rules in the ρCal should be somehow simulated in the corresponding *CRS* rewrite system.

We have considered in this paper *CRS* satisfying the pattern condition. However, we believe that the results obtained can be applied also to general *CRS* and a similar correspondence between *CRS*-reductions and reductions in an appropriate version of ρCal can be defined similarly.

Other higher order rewrite systems have already been compared, for example the *CRS* and the *Higher-order Rewrite Systems* (*HRS*) [18,19]. The detailed comparison can be found in [22] and reveals that the two systems have the same expressive power and therefore they can be considered equivalent. Using this comparison, we can have an indirect representation of the *HRS* in the $\rho Cal_{\P}$ composing the translation from the *HRS* to the *CRS* as defined in [22] with the translation from *CRS* to ρCal we have defined in this paper. Some other higher order systems like the Expression Reduction Systems of Khasidashvili [13] (*ERS*) should be considered. Although *CRS* and *ERS* are conceptually very similar, their syntax differs in many aspects (for example the restriction in the *ERS* to *admissibles* assignments). Therefore, it may be interesting to analyze also the correspondence between the ρCal and the *ERS* or other systems like the Explicit Reduction Systems of Pagano [21].

Acknowledgements.

The authors wish to thank Luigi Liquori for fruitful discussions and comments. We would also like to thank Femke van Raamsdonk for providing an important list of CRS examples.

References

[1] F. Barbanera, M. Fernández, and H. Geuvers. Modularity of Strong Normalisation and Confluence in the Algebraic λ-Cube. *Journal of Functional Programming*, 7(6):613–660, November 1997.

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

[3] 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.

[4] C. Bertolissi, H. Cirstea, and C. Kirchner. Translating Combinatory Reduction Systems into the Rewriting Calculus. Technical Report A03-R-057, LORIA, June 2003. `http://www.loria.fr/~bertolis/A03-R-057.ps`.

[5] F. Blanqui. *Type Theory and Rewriting*. PhD thesis, University Paris-Sud, 2001.

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

[7] 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.

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

[9] H. Cirstea, C. Kirchner, and L. Liquori. The Rho Cube. In *Proc. of FOSSACS*, volume 2030 of *LNCS*, pages 166–180, 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), September 2002. Electronic Notes in Theoretical Computer Science.

[11] G. Faure and C. Kirchner. Exceptions in the Rewriting Calculus. In *Proc. of RTA*, volume 2378 of *LNCS*, pages 66–82. Springer-Verlag, 2002.

[12] J. Jouannaud and M. Okada. Abstract Data Type Systems. *Theoretical Computer Science*, 173(2):349–391, 1997.

[13] Z. Khasidashvili. Expression reduction systems. In *Proceedings of I. Vekua Institute of Applied Mathematics*, volume 36, pages 200–220, 1990.

[14] Z. Khasidashvili. Church-rosser theorem in orthogonal reduction systems. Technical report, INRIA Rocquencourt, 1992.

[15] J. Klop, V. v. Oostrom, and F. v. Raamsdonk. Combinatory Reduction Systems: Introduction and Survey. *Theoretical Computer Science*, 121:279–308, 1993. Special issue in honour of Corrado Böhm.

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

[17] D. Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. In *Proc. of ELP*, volume 475 of *LNCS*, pages 253–281. Springer-Verlag, 1991.

[18] T. Nipkow. Higher-order critical pairs. In *Proceedings of Logic in Computer Science*, pages 342–349, 1991.

[19] T. Nipkow. Orthogonal higher-order rewrite systems are confluent. In *Proceedings of the International Conference on Typed La mbda Calculi and Applications*, pages 306–317, 1993.

[20] T. Nipkow and C. Prehofer. Higher-Order Rewriting and Equational Reasoning. In W. Bibel and P. Schmitt, editors, *Automated Deduction — A Basis for Applications. Volume I: Foundations*. Kluwer, 1998.

[21] B. Pagano. *Des calculs de substitution explicites et de leur application à la compilation des langages fonctionnels*. PhD thesis, U. Paris VI, 1997.

[22] V. van Oostrom and F. van Raamsdonk. Comparing combinatory reduction systems and higher-order rewrite systems. Report CS-R9361, CWI, september 1993.

[23] D. A. Wolfram. *The Clausal Theory of Types*, volume 21 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1993.