

# Translating Combinatory Reduction Systems into the Rewriting Calculus

Clara Bertolissi, Horatiu Cirstea, Claude Kirchner

► **To cite this version:**

Clara Bertolissi, Horatiu Cirstea, Claude Kirchner. Translating Combinatory Reduction Systems into the Rewriting Calculus. 4th International Workshop on Rule-Based Programming - RULE 2003, 2003, Valencia, Spain, 2003. <inria-00107640>

**HAL Id: inria-00107640**

**<https://hal.inria.fr/inria-00107640>**

Submitted on 19 Oct 2006

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

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

Clara Bertolissi, 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

www.loria.fr/{~bertolis,~cirstea,~ckirchne}

**Abstract.** The last few years have seen the development of the *rewriting calculus* (or rho-calculus,  $\rho Cal$ ) that extends first order term rewriting and  $\lambda$ -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  $\lambda$ -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 groundbreaking 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  $\lambda$ -calculus algebraic features. In the first case, we find the works on CRS [16] and other higher-order rewriting systems [23,20], in the second case the works on combination of  $\lambda$ -calculus with term rewriting [2,5,13] to mention only a few.

For example, the *Combinatory Reduction Systems (CRS)*, introduced by J.W.Klop [15] are an extension of first order rewrite systems with a mechanism of bound variables like in the  $\lambda$ -calculus. The meta-language of *CRS*, *i.e.* the language in which the notions of substitution and rewrite step are expressed, is based on  $\lambda$ -calculus and higher-order matching (matching modulo the  $\beta$ -rule).

In the same line, the Rewriting Calculus (also called Rho Calculus or  $\rho Cal$ ) [7,8,10], extends first order term rewriting and  $\lambda$ -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  $\rho Cal$  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  $\rho Cal$ , and, using this translation, the proof that every reduction of a *CRS*-term can be reproduced in the  $\rho Cal$ .

The paper is structured as follows: In Section 2 we briefly present the  $\rho Cal$  through its components. In Section 3 we give a description of *CRSs* and we give some examples. In Section 4 we present a translation from *CRS*-terms into  $\rho$ -terms and we state a theorem on the correspondence between *CRS*-reductions and  $\rho$ -reductions. Section 5 concludes the paper and gives some perspectives to this work. The detailed proofs as well as some more examples are given in the Appendix.

## 2 The $\rho Cal$

We briefly present in what follows the syntax and the semantics of the  $\rho Cal$ . For a more detailed presentation the reader can refer to [8,10] and for a typed version of the calculus to [4,9].

### 2.1 Syntax

In this paper, the symbols  $t, u, \dots$  range over the set  $\mathcal{T}$  of terms, the symbols  $X, Y, Z, \dots, x, y, z \dots$  range over the infinite set  $\mathcal{X}$  of variables (we usually use

capitals for free variables) and the symbols  $f, g, \dots$  range over the infinite set  $\mathcal{F}$  of constants. All symbols can be indexed. The set of  $\rho$ -terms is defined as follows:

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

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 “ $\rightarrow$ ” which is, in turn, of higher priority than the “,”. By abuse of notation, function application is denoted by  $t_0(t_1 \dots t_n)$  instead of  $t_0 t_1 \dots t_n$ .

To support the intuition, we mention here that the application of an abstraction  $t_1 \rightarrow t_3$  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  $\sigma$  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 “ $\alpha$ -convention” of Church [6], and modulo the “*hygiene-convention*” of Barendregt [3].

We should mention that there are two operators binding variables: in  $t_1 \rightarrow t_2$ , 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  $t_2$ .

**Definition 1 (Free and bound variables).**

$$\begin{aligned} \mathcal{FV}(f) &\triangleq \{f\} & \mathcal{FV}(t_1 \rightarrow t_2) &\triangleq \mathcal{FV}(t_2) \setminus \mathcal{FV}(t_1) \\ \mathcal{FV}(X) &\triangleq \{X\} & \mathcal{FV}([t_1 \ll t_2].t_3) &\triangleq (\mathcal{FV}(t_3) \setminus \mathcal{FV}(t_1)) \cup \mathcal{FV}(t_2) \\ \mathcal{FV}(t_1 t_2) &\triangleq \mathcal{FV}(t_1) \cup \mathcal{FV}(t_2) & \mathcal{FV}(t_1, t_2) &\triangleq \mathcal{FV}(t_1) \cup \mathcal{FV}(t_2) \end{aligned}$$

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.

*Example 1.* We list some examples of  $\rho$ Cal-terms:

- $x \rightarrow x$  corresponds to the  $\lambda$ -term  $\lambda x.x$ ;
- $x \rightarrow 0, y \rightarrow 0$  denotes a record with two fields,  $x$  and  $y$ ;
- $(f(x) \rightarrow x) f(a)$  represents the application of the rewrite rule  $f(x) \rightarrow x$  to the term  $f(a)$ ;
- $[f(x) \ll f(a)].x$  represents the delayed matching constraint obtained when evaluating the application above.

## 2.2 Matching

The evaluation mechanism of the calculus relies on the *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, a syntactic theory, an equational theory or even more elaborated (higher-order matching) theories [8].

First, the classical notion of simultaneous substitution application is adapted to deal with the new forms of constrained terms introduced in the  $\rho\text{Cal}$ .

**Definition 2 (Substitutions).**

A substitution  $\sigma$  is a mapping from the set of variables to the set of terms. A finite substitution has the form  $\sigma = \{x_1/t_1 \dots x_m/t_m\}$  where  $\text{Dom}(\sigma) = \{x_1, \dots, x_m\}$ ; the empty substitution  $\{\}$  is denoted by  $\sigma_{\text{ID}}$ . The application of a substitution  $\sigma$  to a term  $t$ , denoted by  $\sigma(t)$  or  $t\sigma$ , is defined as follows:

$$\begin{array}{ll} \sigma_{\text{ID}}(t) \triangleq t & \sigma(t_1 \rightarrow t_2) \triangleq \sigma(t_1) \rightarrow \sigma(t_2) \\ \sigma(f) \triangleq f & \sigma([t_1 \ll t_2].t_3) \triangleq [\sigma(t_1) \ll \sigma(t_2)].\sigma(t_3) \\ \sigma(x_i) \triangleq \begin{cases} t_i & \text{if } x_i \in \text{Dom}(\sigma) \\ x_i & \text{otherwise} \end{cases} & \sigma(t_1 t_2) \triangleq \sigma(t_1) \sigma(t_2) \\ & \sigma(t_1, t_2) \triangleq \sigma(t_1), \sigma(t_2) \end{array}$$

This defines higher-order substitutions as we work modulo  $\alpha$ -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. When implicit renaming is not assumed a more complex definition should be used or alternatively, explicit substitutions can be considered [7].

We assume given a theory  $\mathbb{T}$ , defined for example equationally, and we define matching problems in this general setting.

**Definition 3 (Matching).** Given a theory  $\mathbb{T}$  on the  $(\rho)$ -terms, a  $\mathbb{T}$ -match-equation is a formula of the form  $t_1 \prec_{\mathbb{T}} t_2$ , where  $t_1$  and  $t_2$  are two  $\rho$ -terms. A substitution  $\sigma$  is solution of the match-equation  $t_1 \prec_{\mathbb{T}} t_2$  if  $\mathbb{T} \models \sigma(t_1) = t_2$ . A  $\mathbb{T}$  matching-system is a conjunction of  $\mathbb{T}$  match-equations. A substitution is solution of a  $\mathbb{T}$  matching-system  $P$  if it is solution of all the  $\mathbb{T}$ -match-equations in  $P$ . We denote by  $\text{Sol}(P)$  the set of all solutions of  $P$ .

Since we can consider arbitrary theories over  $\rho$ -terms,  $\mathbb{T}$ -matching is in general undecidable. We are interested here in the decidable cases. If the underlying matching theory is clear from the context (usually when using a syntactic theory) then it is omitted from the notation.

**2.3 Semantics**

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

$$\begin{aligned}
(\rho) \quad & (t_1 \rightarrow t_2)t_3 \rightarrow_\rho [t_1 \ll t_3].t_2 \\
(\sigma) \quad & [t_1 \ll t_3].t_2 \rightarrow_\sigma \sigma_1(t_2), \dots, \sigma_n(t_2), \dots \\
& \text{where } \sigma_i \in \text{Sol}(t_1 \ll_{\mathbb{T}} t_3) \\
(\delta) \quad & (t_1, t_2)t_3 \rightarrow_\delta t_1 t_3, t_2 t_3
\end{aligned}$$

**Fig. 1.** Small-step reduction semantics

the form  $\sigma = \{x_1/t_1 \dots x_m/t_m\}$  where  $\text{Dom}(\sigma) = \{x_1, \dots, x_m\}$ . The application of a substitution  $\sigma$  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  $\rho\text{Cal}$  (see [10] and Definition 2 in Appendix).

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

$$(\text{Ctx}[-]) \quad \frac{t_1 \rightarrow_{\rho\delta} t_2}{\text{Ctx}[t_1] \mapsto_{\rho\delta} \text{Ctx}[t_2]} \quad \forall t_1, t_2 \in \mathcal{T}$$

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

*Example 2 (Small-step reductions).* We take the terms

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

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  $\mapsto_{\rho\delta}$ ):

1.  $(f(\mathcal{X}) \rightarrow (3 \rightarrow 3)\mathcal{X})f(3) \mapsto_\rho \underline{(f(\mathcal{X}) \rightarrow [3 \ll \mathcal{X}].3)}f(3) \mapsto_\rho$   
 $\underline{[f(\mathcal{X}) \ll f(3)].([3 \ll \mathcal{X}].3)} \mapsto_\sigma [3 \ll 3].3 \mapsto_\sigma 3$
2.  $(f(\mathcal{X}) \rightarrow (3 \rightarrow 3)\mathcal{X})f(4) \mapsto_\rho \underline{[f(\mathcal{X}) \ll f(4)].((3 \rightarrow 3)\mathcal{X})} \mapsto_\sigma$   
 $\underline{(3 \rightarrow 3)4} \mapsto_\rho [3 \ll 4].3$

## 2.4 The $\rho\text{Cal}_\lambda$ and the $\rho\text{Cal}_\mathbb{T}$

The general  $\rho\text{Cal}$  can be instantiated to simpler versions when the syntax is restricted and the matching theory used in the  $(\sigma)$  rule is specified [7]. For example, when all the  $\rho$ -rules are of the form  $\mathcal{X} \rightarrow \mathcal{T}$  and a syntactic matching is used, a version similar to the lambda-calculus, denoted  $\rho\text{Cal}_\lambda$ , is obtained. We denote the reductions in the obtained calculus by  $\rightarrow_{\rho_\lambda}$ . 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  $\rho Cal$  can be obtained using more elaborated (decidable) matching theories like, for example, pattern matching [17]. Our goal is to define a version of the  $\rho Cal$  with an evaluation mechanism similar to that of CRSs and thus using a (pattern) higher-order matching. Therefore, we introduce now the notion of  $\rho$ -pattern directly inspired from the  $CRS$ -patterns (defined in Section 3).

**Definition 4 ( $\rho$ -pattern).** *A  $\rho$ -term  $p$  is called a  $\rho$ -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,  $x \rightarrow f(Z x)$  is a  $\rho$ -pattern while  $x \rightarrow (Z x x)$  and  $g(x \rightarrow x, Z x)$  are not.

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

### 3 Combinatory Reduction Systems

The *Combinatory Reduction Systems (CRS)* have been introduced by J.W.Klop in 1980 to generalise the  $\lambda(a)$ -reductions of Hindley [12] and the contraction schemas of Aczel [1].  $CRS$  are a generalisation of first-order term rewrite systems with bound variables, so as to include both first-order term rewrite systems such as Curry's Combinatory Logic and rewrite systems with bound variables such as Church's  $\lambda$ -calculus.

#### 3.1 Syntax

For defining the components of a  $CRS$  we refer to [16].

In what follows the symbols  $A, B, \dots L, P, R, \dots$  range over the set  $\mathcal{MT}_{CRS}$  of metaterms, the symbols  $t, u, \dots$  range over the set  $\mathcal{T}_{CRS}$  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. We denote by  $\mathcal{F}_m$  ( $\mathcal{Z}_m$ ) the set of  $m$ -ary functional symbols (metavariables). All symbols can be indexed. We denote by  $\equiv$  the syntactic identity of metaterms or substitutions.

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

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

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

We adhere mostly to the notation introduced by Klop [15]. The most notable change is the use of the functional format instead of the applicative one of [15]. 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  $\lambda$ -abstraction of the  $\lambda$ -calculus such that, in  $[x]t$ , the variable  $x$  is bound in  $t$ . The variables bound by  $[\_]$  may be renamed by  $\alpha$ -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.

*Example 3 (Terms and Metaterms).*

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

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 5 (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  $Z(x_1, \dots, x_n)$  where the variables  $x_1, \dots, x_n$ ,  $n \geq 0$ , are distinct and all bound in  $P$ .

This restriction ensures the decidability and the uniqueness of the solution of the matching inherent to the application of the *CRS*-rules. Moreover, the usual conditions used in first-order rewriting are imposed:

**Definition 6.** The set  $\mathcal{R}$  of *CRS* rewrite rules is composed by rules of the form  $L \rightarrow R$  satisfying the following conditions:

- $L$  and  $R$  are closed metaterms
- $L$  has the form  $f(A_1, \dots, A_n)$  with  $A_1, \dots, A_n$  metaterms
- $\mathcal{MV}(L) \supseteq \mathcal{MV}(R)$
- $L$  is a *CRS*-pattern.

A rewrite rule is a couple of metaterms, but the objects which are rewritten are terms. Metavariables permit the definition of reduction schemes since they can be instantiated with the value of all possible terms and thus generate a rule.

*Example 4 ( $\beta$ -rule in *CRS*).* The  $\beta$ -rule of  $\lambda$ -calculus  $(\lambda x.t)u \rightarrow_{\beta} t\{x/u\}$  corresponds in *CRS* to the rewrite rule *BetaCRS*:

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

where  $App \in \mathcal{F}_2$  and  $Ab \in \mathcal{F}_1$  are the encodings for the application operator and the abstraction operator respectively.



### 3.2 Semantics

Next we define in which way a rewrite rule is instantiated to obtain an actual rewrite step. We have to plug in some term in the “holes” of the rule. In *CRS* the holes in the rules are the metavariable. First of all, we need the notions of substitute and assignment.

### 3.3 Substitute and assignment

Given a rewrite rule  $L \rightarrow R$  and a substitution  $\sigma$  (also called assignment, as defined below), we have  $\sigma(L) \mapsto_{L \rightarrow R} \sigma(R)$  if  $\sigma L, \sigma R \in \mathcal{T}_{CRS}$ . 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  $\mapsto_{L \rightarrow R}$  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  $\mapsto_{\mathcal{R}}$  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  $\underline{\lambda}$ -abductor operator are avoided using  $\alpha$ -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.

#### Definition 7 (Substitute).

A  $n$ -ary substitute is an expression of the form  $\xi = \underline{\lambda}x_1 \dots x_n.u$  where  $x_1, \dots, x_n$  are distinct variables and  $u$  is a *CRS*-term (without metavariables). The variables  $x_1, \dots, x_n$  are considered bound by  $\underline{\lambda}$  and may be renamed by  $\alpha$ -conversion.

A substitute  $\underline{\lambda}x_1 \dots x_n.u$  can be applied to an  $n$ -tuple of *CRS*-terms  $(t_1, \dots, t_n)$ , yielding to the simultaneous substitution of  $x_1, \dots, x_n$  by  $t_1, \dots, t_n$  in  $u$ :

$$(\underline{\lambda}x_1 \dots x_n.u)(t_1, \dots, t_n) \triangleq u\{x_1 := t_1, \dots, x_n := t_n\}$$

**Definition 8 (Assignment).** An assignment  $\sigma = \{(Z_1, \xi_1), \dots, (Z_n, \xi_n)\}$ , is a finite set of pairs (metavariable, substitute) such that  $\text{arity}(Z_i) = \text{arity}(\xi_i) \forall i \in \{1, \dots, n\}$ . The application of an assignment  $\sigma$  to a *CRS*-metaterm  $t$ , denoted  $\sigma(t)$  or  $t\sigma$ , is inductively defined in the following way:

$$\begin{aligned} \sigma([x]t) &\triangleq [x]\sigma(t) & \sigma(f(t_1, \dots, t_n)) &\triangleq f(\sigma(t_1), \dots, \sigma(t_n)) \\ \sigma(Z_i) &\triangleq \xi_i \quad \text{if } (Z_i, \xi_i) \in \sigma & \sigma(Z_i(t_1, \dots, t_n)) &\triangleq \sigma(Z_i)(\sigma(t_1), \dots, \sigma(t_n)) \downarrow_{\underline{\beta}} \end{aligned}$$

A variable in an instance of a metavariable should be bound only if it is bound in the occurrence of the metavariable. Unintended bindings can be avoided by renaming bound variables. In the following we will assume that this is done whenever necessary. To simplify the notation, we will often write  $\sigma t$  instead of  $\sigma(t)$ .

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  $\lambda$ -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.

For example, if  $t = Z \in \mathcal{Z}_n$  and  $\sigma = \{\dots, (Z, \xi), \dots\}$  where  $\xi = \underline{\lambda}x_1 \dots x_n.u$  then

$$\begin{aligned} \sigma(Z(t_1, \dots, t_n)) &\triangleq \sigma(Z)(\sigma(t_1), \dots, \sigma(t_n)) \downarrow_{\underline{\beta}} \\ &\triangleq (\underline{\lambda}x_1 \dots x_n.u)(\sigma(t_1), \dots, \sigma(t_n)) \downarrow_{\underline{\beta}} \\ &\triangleq u\{x_1 := \sigma(t_1), \dots, x_n := \sigma(t_n)\}. \end{aligned}$$

*Example 5.* 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 4). 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 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  $t \mapsto_{L \rightarrow R} f(a)$  and thus  $f(t) \mapsto_{L \rightarrow R} f(f(a))$ .

A *CRS* in which every metavariable has an arity equal to zero and no rewrite rule contains the abstraction operator  $[-]$  corresponds to a first-order rewrite system.

In fact, in *CRS* the higher order substitution is necessary when we have an assignment applied to a metavariable, i.e. when we could have the meta-lambda working. If the metavariable has an arity equal to zero, we do not have higher order substitution. The relation generated by the *CRS* rewrite system is identical to the one generated by a first-order rewrite system.

*Example 6.* Given a first-order rewrite system describing the addition on natural numbers, we can express it by the *CRS* with

- $\mathcal{Z} = \{X, Y, Z, \dots\}$  all of arity 0
- $\mathcal{F} = \{s, a, 0\}$

where  $s \in \mathcal{F}_1$  is the successor operator,  $a \in \mathcal{F}_2$  the addition operator and  $0 \in \mathcal{F}_0$  the zero of naturals; and the set  $\mathcal{R}$  of rewriting rules:

- $\mathcal{R} = \{R_1 : a(0, X) \rightarrow X, R_2 : a(s(X), Y) \rightarrow s(a(X, Y))\}$

Let  $t = a(s(0), s(s(x)))$  be a *CRS*-term. We can reduce  $t$  by applying first the rule  $R_2$  and then the rule  $R_1$ . Using the assignment  $\sigma = \{(X, 0), (Y, s(s(x)))\}$  we obtain

$$a(s(0), s(s(x))) \rightarrow_{R_1} s(a(0, s(s(x))))$$

and then apply the rule  $R_1$  using the assignment  $\sigma = \{(X, s(s(x)))\}$  and we obtain

$$s(a(0, s(s(x)))) \rightarrow_{R_2} s(s(s(x)))$$

## 4 Translating the *CRS* into the Rewriting Calculus

We propose in this section a translation of *CRS*-(meta)terms into  $\rho$ -terms and we show that to *CRS*-reductions correspond  $\rho$ -reductions. The assignment application used for performing term reductions in a *CRS* (and thus the matching the *CRS*-reduction relies on) is based on  $\lambda$ -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  $\rho\mathit{Cal}_{\mathfrak{q}}$  as target calculus.

In the following we suppose that the set of constants of the considered  $\rho\mathit{Cal}_{\mathfrak{q}}$  contains the set of functional symbols of the corresponding *CRS* and the set of variables of  $\rho\mathit{Cal}_{\mathfrak{q}}$  contains the variables and metavariables of the corresponding *CRS*.

**Definition 9 (Translation).** *The translation of a *CRS*-metaterm  $t$  into a  $\rho$ -term, denoted  $\bar{t}$ , is inductively defined as follows:*

- *CRS-terms into  $\rho$ -terms*

$$\begin{array}{ll} \bar{x} \triangleq x & \overline{f(t_1, \dots, t_n)} \triangleq f(\bar{t}_1 \dots \bar{t}_n) \\ \overline{[x]t} \triangleq x \rightarrow \bar{t} & \overline{Z(t_1, \dots, t_n)} \triangleq Z \bar{t}_1 \dots \bar{t}_n \end{array}$$

- *CRS rewrite rules into  $\rho$ -terms:*

$$\overline{L \rightarrow R} \triangleq \bar{L} \rightarrow \bar{R}$$

- *CRS substitutes into  $\rho$ -terms:*

$$\overline{\lambda x_1 \dots x_n. u} \triangleq x_1 \rightarrow (x_2 \rightarrow (\dots (x_n \rightarrow \bar{u}) \dots))$$

- *CRS assignments into  $\rho$ -substitutions:*

$$\overline{\{\dots, (Z, \lambda x_1 \dots x_n. u), \dots\}} \triangleq \{\dots, Z / x_1 \rightarrow (x_2 \rightarrow (\dots (x_n \rightarrow \bar{u}) \dots)), \dots\}$$

We can observe that the translation of the *CRS*-abstraction operator “[ ]” corresponds to the  $\rho$ -abstraction operator “ $\rightarrow$ ”. An  $n$ -ary *CRS*-metavariable (function) corresponds in the  $\rho\mathit{Cal}_{\mathfrak{q}}$  to a variable (constant) applied to  $n$   $\rho$ -terms.

Since in  $\rho\mathit{Cal}$  rewrite rules are first class objects, a *CRS* rewrite rule is translated into a  $\rho$ -term and more precisely into a  $\rho$ -rule.

The  $\lambda$ -abstraction operator defined at the meta-level of *CRS* is translated into the  $\rho$ -abstraction operator “ $\rightarrow$ ”. This means that reductions performed in *CRS* at the meta-level (using  $\lambda$ ) correspond in the  $\rho\mathit{Cal}_{\mathfrak{q}}$  to explicit reductions corresponding to the application of the abstraction operator “ $\rightarrow$ ”.

The translation of the abstraction operator and of the rewrite rules of a *CRS* into the same abstraction operator of the  $\rho\mathit{Cal}$  corresponds to the uniform treatment of first and higher-order rewriting in the  $\rho\mathit{Cal}$ .

*Example 7.* We have already seen how the  $\beta$ -rule of  $\lambda$ -calculus can be translated into a *CRS*. When translating this *BetaCRS* rule into the rewriting calculus the following term is obtained:

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

where  $\text{App}, \text{Ab} \in \mathcal{F}$  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  $\rho$ -abstraction operator ensuring that the corresponding variables are bound in the translation and thus, the preservation of the pattern condition by the translation.

**Lemma 1.** *Given a CRS-metaterm  $L$ . If  $L$  satisfies the CRS pattern definition, then  $\overline{L}$  satisfies the  $\rho$ -pattern definition.*

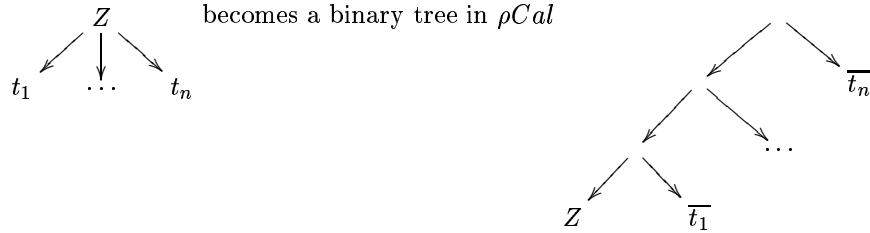
*Proof.* Follows immediately from Definition 9 and Definition 4.

Moreover, as a consequence of the definition of the translation, we have that every  $\rho$ -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  $\rho\text{Cal}_{\mathfrak{q}}$ . The complete proofs as well as some other examples can be found in the Appendix.

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  $\epsilon$  the head position of  $A$ .  $A(\omega)$  is the symbol at the position  $\omega$  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  $A|_{[B]_{\omega}}$  to signify that  $A$  has a sub-metaterm  $B$  at the position  $\omega$ . To simplify the notation we write  $A(\omega, \underbrace{i \dots i}_n, \omega')$  for  $A(\omega, (i \dots i), \omega')$ .

The position  $\omega$  of a sub-metaterm  $\overline{B}$  in a *CRS*-metaterm  $A|_{[B]_{\omega}}$  and the position of its translation  $\overline{B}$  in the  $\rho$ -term  $\overline{A}$  are not the same. This is due to the different use of (meta)variables and functional symbols in the *CRS*- and  $\rho$ -(meta)terms. Indeed, since an  $n$ -ary metavariable and an  $n$ -ary function are translated in the  $\rho\text{Cal}_{\mathfrak{q}}$  as a variable or function applied to  $n$   $\rho$ -terms, the  $n$ -ary tree of the *CRS*-metaterm



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

**Definition 10 (Position transformation).** *Let  $A$  be a CRS-metaterm and  $\omega \in \mathcal{Pos}(A)$  a position in the metaterm  $A$ . The transformation function  $tr(\omega, A) : \mathcal{Pos}(A) \times \mathcal{MTCRS} \mapsto \mathcal{Pos}(\bar{A})$  is inductively defined in the following way:*

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

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

**Lemma 2.** *Let  $A_{[B]_\omega}$  be a CRS-metaterm with a sub-term  $B$  at the position  $\omega$ . Then:*

$$\overline{A_{[B]_\omega}} = \bar{A}_{[\bar{B}]_{tr(\omega, A)}}$$

*Proof.* We use structural induction on the metaterm  $A$ .

- If  $A = x$  then

$$\overline{x_{[B]_\epsilon}} = \bar{x} = \bar{x}_{[\bar{B}]_\epsilon} = \bar{x}_{[\bar{B}]_{tr(\epsilon, x)}}$$

- If  $A = [x]s$  then  $A_{[x]_1[s]_2}$

We note that  $\omega$  is either of the form  $\omega = 1.\epsilon$  or of the form  $\omega = 2.\omega'$  with  $\omega'$  any position in  $s$ . Therefore we have two cases:

- a)  $\omega = 1.\epsilon$  then  $B = x$

$$\overline{([x]s)_{[B]_{1.\epsilon}}} = \overline{[x]s} = \overline{[x]s}_{[x]_{1.\epsilon}} = \overline{[x]s}_{[\bar{x}]_{1.\epsilon}} = \overline{[x]s}_{[\bar{B}]_{tr(1.\epsilon, [x]s)}}$$

- b)  $\omega = 2.\omega'$

$$\begin{aligned} \overline{([x]s)_{[B]_{2.\omega'}}} &= \overline{[x](s_{[B]_{\omega'}})} \\ &= x \rightarrow \bar{s}_{[B]_{\omega'}} \\ &\stackrel{ih}{=} x \rightarrow (\bar{s}_{[\bar{B}]_{tr(\omega', s)}}) \quad (ih = \text{by induction hypothesis}) \\ &= (x \rightarrow \bar{s})_{[\bar{B}]_{2.tr(\omega', [x]s)}} \\ &= \overline{[x]s}_{[\bar{B}]_{tr(2.\omega', [x]s)}} \end{aligned}$$

- If  $A = Z(A_1, \dots, A_n)$  (or  $A = f(A_1, \dots, A_n)$ ) and  $\omega = n - i.\omega'$  where  $i \in \{0, \dots, n-1\}$  then

$$\begin{aligned}
\overline{Z(A_1, \dots, A_n)}_{[B]_{n-i, \omega'}} &= \overline{Z(A_1, \dots, A_{n-i}[B]_{\omega'}, \dots, A_n)} \\
&= \overline{Z(\overline{A_1} \dots \overline{A_{n-i}[B]_{\omega'}} \dots \overline{A_n})} \\
&\stackrel{ih}{=} \overline{Z(\overline{A_1} \dots \overline{A_{n-i}[B]_{tr(\omega', A_{n-i})}} \dots \overline{A_n})} \\
&= \overline{(Z(\overline{A_1} \dots \overline{A_n}))}_{[B]_{1^{i.2}.tr(\omega', A_{n-i})}} \\
&= \overline{Z(A_1, \dots, A_n)}_{[B]_{tr(n-i, \omega', Z(A_1, \dots, A_n))}}
\end{aligned}$$

□

The following proposition shows how *CRS* substitution is naturally translated into the substitution of the rewriting calculus .

**Lemma 3.** *Let  $A$  be CRS-metaterm,  $t_1, \dots, t_n$  be CRS-terms and  $x_1, \dots, x_n \in FV(A)$  be distinct variables. Then*

$$\overline{A\{x_1 := t_1, \dots, x_n := t_n\}} = \overline{A\{x_1/\overline{t_1}, \dots, x_n/\overline{t_n}\}}$$

*Proof.* We use induction on the number of variables in the domain of the substitution.

*Base case :*  $\overline{A\{x := t\}}$ .

Let  $\omega_i, i \in \{1, \dots, m\}$ , be all the occurrences of  $x$  in  $A : A_{[x]_{\omega_1}} \dots [x]_{\omega_m}$ .

We use an induction on  $m$ .

– *Base case :*  $m = 1$ .

$$\begin{aligned}
\overline{A\{x := t\}} &= \overline{A_{[t]_{\omega_1}}} \\
&= \overline{A_{[\overline{t}]_{tr(\omega_1, A)}}} \quad (\text{Lemma 2}) \\
&= \overline{A\{x/\overline{t}\}}.
\end{aligned}$$

– *Induction :* By induction hypothesis on the first  $m - 1$  occurrences of  $x$ :

$$\begin{aligned}
\overline{A\{x := t\}} &= \overline{(A_{[t]_{\omega_1}} \dots [t]_{\omega_{m-1}})}_{[t]_{\omega_m}} \\
&= \overline{A_{[t]_{\omega_1}} \dots [t]_{\omega_{m-1}} [\overline{t}]_{tr(\omega_m, A)}} \quad (\text{Lemma 2}) \\
&\stackrel{ih}{=} \overline{(\overline{A}_{[\overline{t}]_{tr(\omega_1, A)}} \dots [\overline{t}]_{tr(\omega_{m-1}, A)})}_{[\overline{t}]_{tr(\omega_m, A)}} \\
&= \overline{A\{x/\overline{t}\}}.
\end{aligned}$$

Now we use induction to prove the case we have  $n$  distinct variables to substitut in  $A$ .

*Induction :*  $\overline{A\{x_1 := t_1, \dots, x_n := t_n\}}$ .

By induction hypotesis we have:

$$\overline{A\{x_1 := t_1, \dots, x_{n-1} := t_{n-1}\}} = \overline{A\{x_1/\overline{t_1}, \dots, x_{n-1}/\overline{t_{n-1}}\}}$$

Let  $w := A\{x_1 := t_1, \dots, x_{n-1} := t_{n-1}\}$  then we have  $\overline{w\{x_n := t_n\}}$  and we are again in the base case.

□

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

**Lemma 4.** *Let  $A$  be a CRS-metaterm and  $\sigma$  an assignment. Then*

$$\overline{\sigma(A)} \mapsto_{\rho_\lambda} \overline{\sigma(A)}$$

*Proof.* We use a structural induction on the metaterm  $A$ .

– If  $A$  is a variable  $x$ , then

$$\begin{aligned} \overline{\sigma(x)} &= \overline{\sigma(x)} \\ &= x \quad (\text{since } x \notin \text{Dom}(\overline{\sigma}) \text{ because } x \notin \mathcal{MV}) \\ &= \overline{x} \\ &= \overline{\sigma(x)} \end{aligned}$$

– If  $A = [x]s$  then

$$\begin{aligned} \overline{\sigma([x]s)} &= \overline{\sigma(x \rightarrow \overline{s})} \\ &= x \rightarrow (\overline{\sigma(\overline{s})}) \\ &\mapsto_{\rho_\lambda} x \rightarrow \overline{\sigma(s)} \quad (ih) \\ &= \overline{[x]\sigma(s)} \\ &= \overline{\sigma([x]s)} \end{aligned}$$

– If  $A = f(A_1, \dots, A_n)$  then

$$\begin{aligned} \overline{\sigma(f(A_1, \dots, A_n))} &= \overline{\sigma(f(\overline{A_1} \dots \overline{A_n}))} \\ &= f(\overline{\sigma(A_1)} \dots \overline{\sigma(A_n)}) \\ &\mapsto_{\rho_\lambda} f(\overline{\sigma(A_1)} \dots \overline{\sigma(A_n)}) \quad (ih) \\ &= \overline{f(\sigma(A_1), \dots, \sigma(A_n))} \\ &= \overline{\sigma(f(A_1, \dots, A_n))} \end{aligned}$$

– If  $A = Z(A_1, \dots, A_n)$  then we use induction on the arity of  $Z$ :

- $Z$  of arity equal to zero,  $\sigma$  assignment such that  $\sigma = \{\dots, (Z, \xi), \dots\}$  and  $\xi$  contains no  $\underline{\beta}$ -redex.

$$\begin{aligned} \overline{\sigma(Z)} &= \overline{\sigma(Z)} \\ &= Z\{Z/\overline{\xi}\} \\ &= \overline{\xi} \\ &= \overline{\sigma(Z)} \end{aligned}$$

- $Z$  of arity  $n$ ,  $\sigma$  assignment such that  $\sigma = \{\dots, (Z, \xi), \dots\}$  and  $\xi$  of the form  $\lambda x_1 \dots x_n. u$ . We suppose that  $A_1, \dots, A_n$  have no occurrences of  $x_1 \dots x_n$  which is always possible by  $\alpha$ -conversion.

$$\begin{aligned} \overline{\sigma(Z(A_1, \dots, A_n))} &= \overline{\sigma(Z \overline{A_1} \dots \overline{A_n})} \\ &= ((\dots ((\overline{\sigma(Z)}) \overline{\sigma(A_1)}) \dots) \overline{\sigma(A_n)}) \\ &= ((\dots (Z\{Z/ x_1 \rightarrow (x_2 \rightarrow (\dots (x_n \rightarrow \overline{u}) \dots))\} \overline{\sigma(A_1)}) \dots) \overline{\sigma(A_n)}) \\ &= ((\dots ((x_1 \rightarrow (x_2 \rightarrow (\dots (x_n \rightarrow \overline{u})))) \overline{\sigma(A_1)}) \dots) \overline{\sigma(A_n)}) \\ &\mapsto_{\rho_\lambda} (\dots (\overline{u} \{x_1/\overline{\sigma(A_1)}\}) \dots) \{x_n/\overline{\sigma(A_n)}\} \\ &= \overline{u \{x_1/\overline{\sigma(A_1)}, \dots, x_n/\overline{\sigma(A_n)}\}} \end{aligned}$$

$$\begin{aligned}
& (\text{since } \forall i \ x_i \text{ not in } A_1, \dots, A_n). \\
& \mapsto_{\rho_\lambda} \overline{u} \{x_1/\overline{\sigma(A_1)}, \dots, x_n/\overline{\sigma(A_n)}\} \text{ (ih)} \\
& = u\{x_1 := \sigma(A_1), \dots, x_n := \sigma(A_n)\} \text{ (By Lemma 3)} \\
& = (\lambda x_1 \dots x_n. u)(\sigma(A_1), \dots, \sigma(A_n)) \downarrow_{\underline{\beta}} \\
& = \overline{(\sigma(Z))(\sigma(A_1), \dots, \sigma(A_n))} \\
& = \overline{\sigma(Z(A_1, \dots, A_n))}
\end{aligned}$$

□

In the last case concerning the  $n$ -ary metavariables, we need to add a supplementary constraint on  $A_1, \dots, A_n$  so that the composition of substitutions  $\{x_1/\overline{\sigma(A_1)}\}, \dots, \{x_n/\overline{\sigma(A_n)}\}$  becomes equal to the simultaneous corresponding substitution.

As an immediate consequence we obtain as well  $\overline{\sigma(\overline{A})} \mapsto_{\rho\sigma} \overline{\sigma(A)}$  and  $\overline{\sigma(\overline{A})} =_{\rho_\lambda} \overline{\sigma(A)}$ . This result is important since  $\mathbb{T}_\lambda$  is the matching theory of the  $\rho\text{Cal}_\mathfrak{q}$  and thus, the  $\mapsto_{\rho_\lambda}$  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  $\text{CRS}$ -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  $\text{CRS}$ -reduction and these steps correspond to explicit  $\rho$ -reductions.

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

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

**Theorem 1.** *Let  $t_0, t_1$  be two  $\text{CRS}$ -terms,  $L \rightarrow R$  a  $\text{CRS}$ -rule and  $\sigma$  an assignment such that  $t_0[\sigma(L)]_\omega$  and  $t_1 \equiv t_0[\sigma(R)]_\omega$ . Given the  $\rho$ -term  $u_0 \triangleq \overline{t_0[x]_{t_r(\omega, t_0)}} \rightarrow \overline{t_0[\overline{L \rightarrow R} x]_{t_r(\omega, t_0)}}$ , then, every derivation resulting from  $u_0 \overline{t_0}$  terminates and converges to  $\overline{t_1}$ :*

$$(u_0 \overline{t_0}) \mapsto_{\rho\sigma\mathfrak{q}} \overline{t_1}$$

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

$$\begin{aligned}
(u_0 \overline{t_0}) & \triangleq (x \rightarrow ((\overline{L \rightarrow R}) x)) \overline{t_0} \triangleq (x \rightarrow ((\overline{L \rightarrow R}) x)) \overline{\sigma(L)} \\
& \triangleq (x \rightarrow ((\overline{L \rightarrow R}) x)) \overline{\sigma(L)} \text{ (By Definition 9)} \\
& \mapsto_\rho [x \ll \overline{\sigma(L)}].((\overline{L \rightarrow R}) x) \\
& \mapsto_\sigma (\overline{L \rightarrow R}) \overline{\sigma(L)}
\end{aligned}$$



$$\begin{aligned}
& \mapsto_{\rho} [\overline{L} \ll \overline{\sigma(L)}].\overline{R} \\
& = [\overline{L} \ll \overline{\sigma(\overline{L})}].\overline{R} \text{ (By Lemma 4)} \\
& \mapsto_{\sigma} \overline{\sigma(\overline{R})} \\
& \mapsto_{\rho\sigma} \overline{\sigma(\overline{R})} \text{ (By Lemma 4)} \\
& \triangleq \overline{t_1}
\end{aligned}$$

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, the two corresponding matching problems  $\overline{L} \ll_{\mathfrak{q}} \overline{\sigma(L)}$  and  $\overline{L} \ll_{\mathfrak{q}} \overline{\sigma(\overline{L})}$  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 2 when matching against the translation of the left-hand side of the rule.

Another possible reduction is the following one:

$$\begin{aligned}
& (u_0 \overline{t_0}) \triangleq (x \rightarrow ((\overline{L} \rightarrow \overline{R}) x)) \overline{t_0} \triangleq (x \rightarrow ((\overline{L} \rightarrow \overline{R}) x)) \overline{\sigma(L)} \\
& \triangleq (x \rightarrow ((\overline{L} \rightarrow \overline{R}) x)) \overline{\sigma(L)} \text{ (By Definition 9)} \\
& \mapsto_{\rho} (x \rightarrow [\overline{L} \ll x].\overline{R}) \overline{\sigma(L)} \\
& \mapsto_{\rho} [x \ll \overline{\sigma(L)}].([\overline{L} \ll x].\overline{R}) \\
& \mapsto_{\sigma} [\overline{L} \ll \overline{\sigma(L)}].\overline{R} \\
& = [\overline{L} \ll \overline{\sigma(\overline{L})}].\overline{R} \text{ (By Lemma 4)} \\
& \mapsto_{\sigma} \overline{\sigma(\overline{R})} \\
& \mapsto_{\rho\sigma} \overline{\sigma(\overline{R})} \text{ (By Lemma 4)} \\
& \triangleq \overline{t_1}
\end{aligned}$$

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

We can notice that we have a longer derivation scheme in the  $\rho\text{Cal}_{\mathfrak{q}}$  than in the *CRS*. For every rewrite step in the *CRS*, in the  $\rho\text{Cal}_{\mathfrak{q}}$  we have two ( $\rho$ )-rule steps plus two ( $\sigma$ )-rule steps for the application of the rewrite rule and some additional steps corresponding to the  $\beta$ -reduction steps performed at the meta-level of the *CRS*-reduction. These latter steps are performed at the object-level of the  $\rho\text{Cal}_{\mathfrak{q}}$  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  $\rho\text{Cal}_{\mathfrak{q}}$  may involve some derivations but this time performed in  $\rho\text{Cal}_{\lambda}$ .

We can generalize the theorem above and build, using the derivations of a term  $t_0$  in a *CRS*, a  $\rho$ -term with a reduction similar to the one of  $t_0$  in the *CRS*.

**Lemma 5.** *Let  $t_0, t_n$  be two *CRS*-terms such that  $t_0 \mapsto_{\mathcal{R}} t_n$  and  $u_0, \dots, u_n$  the corresponding derivation trace terms in the  $\rho\text{Cal}$ . Then every derivation resulting from  $(u_n \dots (u_0 \overline{t_0}))$  terminates and converges to  $\overline{t_n}$ .*

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

If  $t_0 \mapsto_{\mathcal{R}} t_n$  then there exist some *CRS*-terms  $t_1, \dots, t_{n-1}$  and some *CRS*-rewrite rules  $\{L_i \rightarrow R_i\}_{i=0, \dots, n-1} \in \mathcal{R}$  such that

$$t_0 \mapsto_{L_0 \rightarrow R_0} t_1 \mapsto_{L_1 \rightarrow R_1} \dots \mapsto_{L_{n-1} \rightarrow R_{n-1}} t_n$$

We need to build only  $n - 1$   $\rho$ -terms useful for the derivation, *i.e.* such that  $(u_i \overline{t_i}) \mapsto_{\rho\overline{\alpha}_\eta} \overline{t_{i+1}}, \forall i \in \{0, \dots, n - 1\}$ . We can take as last  $\rho$ -term the identity  $u_n = x \rightarrow x$ .

*Base case:*  $u_0 \overline{t_0} \mapsto_{\rho\overline{\alpha}_\eta} \overline{t_n}$  by Theorem 1.

*Induction:*  $u_n (\dots (u_0 \overline{t_0})) \mapsto_{\rho\overline{\alpha}_\eta} \overline{t_n}$ .

We know by induction hypothesis that  $(u_{n-1} (\dots (u_0 \overline{t_0}))) \mapsto_{\rho\overline{\alpha}_\eta} \overline{t_n}$ .

Consequently we have:

$$\begin{aligned} & u_n (\dots (u_1 (u_0 \overline{t_0}))) \\ & \mapsto_{\rho\overline{\alpha}_\eta} u_n \overline{t_n} \text{ (By induction hypothesis)} \\ & \mapsto_{\rho\overline{\alpha}_\eta} (x \rightarrow x) \overline{t_n} \\ & \mapsto_{\rho\overline{\alpha}_\eta} \overline{t_n} \end{aligned}$$

Supposing now evaluating the  $u_n$  redex before having evaluated all the redexes in  $u_{n-1} (\dots (u_1 (u_0 \overline{t_0})))$ . Let us call  $t^*$  the partially evaluated  $\rho$ -term.

$$\begin{aligned} & u_n (\dots (u_1 (u_0 \overline{t_0}))) \\ & \mapsto_{\rho\overline{\alpha}_\eta} u_n t^* \\ & \equiv (x \rightarrow x) t^* \\ & \mapsto_{\rho\overline{\alpha}_\eta} t^* \\ & \mapsto_{\rho\overline{\alpha}_\eta} \overline{t_n} \text{ (By induction hypothesis)} \end{aligned}$$

□

We can state thus that we have a complete and correct translation of *CRS*-reductions to  $\rho$ -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  $\rho$ -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  $\rho\mathcal{C}al$  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  $\beta$ -reduction is implicitly included in the application of the assignment, in the  $\rho\overline{\mathcal{C}al}_\eta$  the corresponding reductions are performed explicitly.

The  $\rho$ -terms  $u_i$  can be built automatically starting from the *CRS*-reduction steps as stated in Theorem 1. 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  $\rho Cal$  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  $\rho Cal$  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 8.* Given the  $CRS$ -reduction  $f(App(Ab([x]f(x)), a)) \mapsto_{\mathcal{R}} f(f(a))$  presented in Example 5. In order to obtain a similar reduction in the  $\rho Cal_{\mathfrak{q}}$ , we consider the  $\rho$ -term  $f(y) \rightarrow f(\overline{BetaCRS} y)$  (with  $BetaCRS$  defined in Example 4) and we apply it to the translation of the initial  $CRS$ -term:

$$\begin{aligned}
& (f(y) \rightarrow f(\overline{BetaCRS} y)) (\overline{f(App(Ab([x]f(x)), a))}) \\
& \mapsto_{\rho} [f(y) \ll f(App(Ab(x \rightarrow f(x)), a))] . f(\overline{BetaCRS} y) \\
& \mapsto_{\sigma} f(\overline{BetaCRS} App(Ab(x \rightarrow f(x)), a)) \\
& \equiv f((App(Ab(x \rightarrow (Z x)), Z_1) \rightarrow Z Z_1) App(Ab(x \rightarrow f(x)), a)) \\
& \mapsto_{\rho} f([App(Ab(x \rightarrow (Z x)), Z_1) \ll App(Ab(x \rightarrow f(x)), a)] . Z Z_1) \\
& \mapsto_{\sigma} f(Z\{Z/z \rightarrow f(z)\} Z_1\{Z_1/a\}) \\
& \quad = f((z \rightarrow f(z)) a) \\
& \mapsto_{\rho} f([z \ll a] . f(z)) \\
& \mapsto_{\sigma} f(f(a))
\end{aligned}$$

One can notice in Example 8 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  $f((z \rightarrow f(z)) a)$  to  $f(f(a))$  are explicit only in the  $\rho Cal$ ). The same behavior is obtained in the following example.

*Example 9* ( $\lambda$ -calculus with surjective pairing).

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

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

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 4.

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  $\sigma = \{(Z, \lambda z. \Pi(\Pi_1 z, \Pi_0 z)), (Z_1, \Pi(x_1, x_2))\}$  and we obtain:

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

Next we apply the rules  $P_1$  and  $P_0$  to the first and second argument of  $\Pi$  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  $\rho\mathit{Cal}_{\mathbf{q}}$ . We translate the set of *CRS* rewrite rules

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

and the *CRS*-term  $t$ :

$$\bar{t} \triangleq \mathit{App}(\mathit{Ab}(z \rightarrow (\Pi(\Pi_1 z, \Pi_0 z))), \Pi(x_1, x_2))$$

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

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

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

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

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

Next, we continue reducing the  $\rho$ -term obtained as intermediary result:

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

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

$$\Pi(\overline{P_1} \Pi(\Pi_1(\Pi(x_1, x_2))), \overline{P_0} \Pi_0(\Pi(x_1, x_2))) \mapsto_{\rho\bar{\sigma}'} \Pi(x_2, x_1)$$

In the following section we will describe some more complicated examples.

## 4.1 More examples

*Example 10 (Sommtation).*

We consider the mathematical finite sommtatory  $\Sigma_{i=0}^{s(n)} s(i)$  where  $n$  is a natural number and  $s$  the successor function.

We can express it in a *CRS* with the following set of rewrite rules  $\mathcal{R}$ :

$$\begin{aligned} \mathcal{R} = \{ & S_0 : a(0, Y) \rightarrow Y ; \\ & S_1 : a(s(X), Y) \rightarrow s(a(X, Y)) ; \\ & Rec_0 : \Sigma(0, [x]F(x)) \rightarrow F(0) ; \\ & Rec_1 : \Sigma(s(N), [x]F(x)) \rightarrow a(\Sigma(N, [x]F(x)), F(s(N))) \} \end{aligned}$$

where  $0 \in \mathcal{F}$  of arity 0,  $s \in \mathcal{F}$  of arity 1,  $\Sigma, a \in \mathcal{F}$  of arity 2,  $X, Y, N \in \mathcal{Z}$  of arity 0 and  $F \in \mathcal{Z}$  of arity 1.

The *CRS*-rules  $S_0$  and  $S_1$  express the addition of natural numbers while the *CRS*-rules  $Rec_0$  and  $Rec_1$  express the sommtatory in a recursive way.

We consider the mathematical sommtatory

$$\Sigma_{i=0}^{s(0)} s(i) = s(0) + s(s(0)) = s(s(s(0)))$$

The correspondent *CRS*-term is  $t = \Sigma(s(0), [i]s(i))$ . Reducing  $t$  we will obtain the same final result.

For performing the reduction of  $t$  we first apply first the rule  $Rec_1$  with the assignment  $\sigma_1 = \{(N, 0), (F, \lambda y.s(y))\}$ . For  $i = 0, 1$ , we denote by  $L_i$  and  $R_i$  the left-hand side and the right-hand side of the rule  $Rec_i$ . Since

$$\begin{aligned} \sigma_1(L_1) &= \sigma_1(\Sigma(s(N), [x]F(x))) \\ &= \Sigma(s(\sigma_1(N)), [x]\sigma_1(F(x))) \\ &= \Sigma(s(0), [x](\lambda y.s(y))(x)) \downarrow_{\underline{\beta}} \\ &= \Sigma(s(0), [x]s(x)) \end{aligned}$$

we have, modulo  $\alpha$ -conversion,  $\sigma_1 L_1 = t$ .

We obtain

$$\begin{aligned} \sigma_1(R_1) &= \sigma_1(a(\Sigma(N, [x]F(x)), F(s(N)))) \\ &= a(\Sigma(\sigma_1(N), [x](\sigma_1(F))(x)), (\sigma_1(F))(s(\sigma_1(N)))) \\ &= a(\Sigma(0, [x](\lambda y.s(y))(x)), (\lambda y.s(y))(s(0))) \downarrow_{\underline{\beta}} \\ &= a(\Sigma(0, [x]s(x)), s(s(0))) \end{aligned}$$

Next we apply the *CRS*-rule  $Rec_0$  to the sub-term  $t_0 = (\Sigma(0, [x]s(x)))$  with the assignment  $\sigma_0 = \{(F, \lambda y.s(y))\}$ .

We verify that  $\sigma_0$  is solution of the matching between  $L_0$  and  $t_0$ , i.e  $\sigma_0(L_0) = t_0$

$$\begin{aligned} \sigma_0(L_0) &= \sigma_0(\Sigma(0, [x]F(x))) \\ &= \Sigma(0, [x]\sigma_0(F(x))) \\ &= \Sigma(0, [x](\lambda y.s(y))(x)) \downarrow_{\underline{\beta}} \\ &= \Sigma(0, [x]s(x)) \end{aligned}$$

We obtain

$$\sigma_0(R_0) = \sigma_0(F(0)) = (\sigma_0(F))(0) = (\underline{\lambda}y.s(y))(0) \rightarrow_{\underline{\beta}} s(0)$$

Therefore, considering the whole *CRS*-term  $t$  we have

$$t \mapsto_{Rec_1} a(\Sigma(0, [x]s(x)), s(s(0))) \mapsto_{Rec_0} a(s(0), s(s(0)))$$

The last steps of the reduction follow easily using the rules  $S_1$  and  $S_0$ :

$$a(s(0), s(s(0))) \mapsto_{S_1} s(a(0, s(s(0)))) \mapsto_{S_0} s(s(s(0)))$$

We translate now the example in the  $\rho Cal_{\eta}$ . We translate the set of *CRS* rewrite rules  $\mathcal{R}$ :

$$\begin{aligned} \overline{S_0} &\triangleq a(0, Y) \rightarrow Y \\ \overline{S_1} &\triangleq a(s(X), Y) \rightarrow s(a(X, Y)) \\ \overline{Rec_0} &\triangleq \Sigma(0, x \rightarrow (F x)) \rightarrow F 0 \\ \overline{Rec_1} &\triangleq \Sigma(s(N), x \rightarrow (F x)) \rightarrow a(\Sigma(N, x \rightarrow (F x)), F s(N)) \end{aligned}$$

and the *CRS*-term  $t$  in a  $\rho$ -term:  $\bar{t} \triangleq \Sigma(s(0), i \rightarrow s(i))$ .

Starting from the *CRS*-reduction shown above, we build the following  $\rho$ -terms corresponding to the application of the  $\overline{Rec_1}$  and  $\overline{Rec_0}$  rules respectively.

$$u_1 = \overline{Rec_1} \text{ and } u_2 = (a(x, y) \rightarrow a(\overline{Rec_0} x, y))$$

and we built the  $\rho$ -term:  $u_2 (u_1 \bar{t})$ .

First of all, we apply to  $\bar{t}$  the  $\rho$ -rule  $\overline{Rec_1}$  with the substitution  $\overline{\sigma_1} = \{N/0, F/y \rightarrow s(y)\}$  and we obtain:

$$\begin{aligned} &\overline{\sigma_1} (a(\Sigma(N, x \rightarrow (F x)), F s(N))) \\ &= a(\Sigma(0, x \rightarrow ((y \rightarrow s(y)) x), (y \rightarrow s(y)) s(0))) \\ &\mapsto_{\rho\sigma} a(\Sigma(0, x \rightarrow s(x)), s(s(0))) \end{aligned}$$

Next, we proceed in the reduction of the  $\rho$ -term obtained as intermediary result:

$$\begin{aligned} &(a(x, y) \rightarrow a(\overline{Rec_0} x, y)) a(\Sigma(0, x \rightarrow s(x)), s(s(0))) \\ &\mapsto_{\rho} [a(x, y) \leftarrow_{\eta} a(\Sigma(0, x \rightarrow s(x)), s(s(0)))].a(\overline{Rec_0} x, y) \\ &\mapsto_{\sigma} a(\overline{Rec_0} \Sigma(0, x \rightarrow s(x)), s(s(0))) \end{aligned}$$

For performing  $\overline{Rec_0}$ , we use the substitution  $\overline{\sigma_2} = \{F/y \rightarrow s(y)\}$  and we obtain

$$\overline{\sigma_2}(F 0) = (y \rightarrow s(y)) 0 \mapsto_{\rho\sigma} s(0)$$

Therefore, considering the whole  $\rho$ -term, we have

$$\bar{t} \mapsto_{\rho\overline{\sigma_1}} a(\Sigma(0, x \rightarrow s(x)), s(s(0))) \mapsto_{\rho\overline{\sigma_2}} a(s(0), s(s(0)))$$

Applying the (first-order) rewrite rules  $\overline{S_0}$  and  $\overline{S_1}$  yields the same final result as in the *CRS*.

$$a(s(0), s(s(0))) \mapsto_{\rho\overline{\sigma_1}} s(a(0, s(s(0)))) \mapsto_{\rho\overline{\sigma_2}} s(s(s(0)))$$

*Example 11 (Recursion).*

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

$$\begin{aligned} \mathcal{R} = \{ & R_0 : \text{rec}(0, A, [xy]F(x, y)) \rightarrow A ; \\ & R_1 : \text{rec}(s(N), A, [xy]F(x, y)) \rightarrow F(N, \text{rec}(N, A, [xy]F(x, y))) \} \end{aligned}$$

where  $\text{rec} \in \mathcal{F}$  of arity 3,  $s \in \mathcal{F}$  of arity 1,  $A, N \in \mathcal{Z}$  of arity 0 and  $F \in \mathcal{Z}$  of arity 2.

We consider the *CRS*-term  $t = \text{rec}(s(0), s(s(0)), [xy]s(y))$  representing the addition of the two natural numbers  $s(0) = 1$  and  $s(s(0)) = 2$ . We reduce the term  $t$  to the final result  $s(s(s(0))) = 3$ .

First of all, we apply to the *CRS*-term  $t$  the *CRS* rewrite rule  $R_1$  with the assignment  $\sigma_1 = \{(N, 0), (A, s(s(0))), (F, \underline{\lambda}z_1z_2.s(z_2))\}$ .

We obtain

$$\begin{aligned} & \sigma_1 (F(N, \text{rec}(N, A, [xy]F(x, y)))) \\ &= (\sigma_1(F))(\sigma_1(N), \text{rec}(\sigma_1(N), \sigma_1(A), [xy]\sigma_1(F(x, y)))) \\ &= (\underline{\lambda}z_1z_2.s(z_2))(0, \text{rec}(0, s(s(0)), [xy](\underline{\lambda}z_1z_2.s(z_2))(x, y))) \downarrow_{\underline{\beta}} \\ &= (\underline{\lambda}z_1z_2.s(z_2))(0, \text{rec}(0, s(s(0)), [xy]s(y))) \downarrow_{\underline{\beta}} \\ &= s(\text{rec}(0, s(s(0)), [xy]s(y))) \end{aligned}$$

Next we apply the rule  $R_0$  to the sub-term  $\text{rec}(0, s(s(0)), [xy]s(y))$  with the assignment  $\sigma_0 = \{(A, s(s(0))), (F, \underline{\lambda}z_1z_2.s(z_2))\}$ .

$$s(\text{rec}(0, s(s(0)), [xy]s(y))) \mapsto_{R_0} s(s(s(0)))$$

We translate the example in the  $\rho\text{Cal}_{\mathfrak{q}}$ . We translate the set of *CRS* rewrite rules  $\mathcal{R}$ :

$$\begin{aligned} \overline{R_0} &\triangleq \text{rec}(0, A, x \rightarrow (y \rightarrow (F \ x \ y))) \rightarrow A \\ \overline{R_1} &\triangleq \text{rec}(s(N), A, x \rightarrow (y \rightarrow (F \ x \ y))) \rightarrow F(N, \text{rec}(N, A, x \rightarrow (y \rightarrow (F \ x \ y)))) \end{aligned}$$

and the *CRS*-term  $t$  in a  $\rho$ -term:

$$\overline{t} \triangleq \text{rec}(s(0), s(s(0)), x \rightarrow (y \rightarrow s(y))).$$

Starting from the *CRS* reduction above, we built the following  $\rho$ -terms corresponding to the application of the  $\overline{R_1}$  and  $\overline{R_0}$  rules respectively.

$$u_1 = \overline{R_1} \text{ and } u_2 = (s(x) \rightarrow s(\overline{R_0} \ x))$$

and we build the  $\rho$ -term:  $u_2 (u_1 \ \overline{t})$ .

First of all, we apply the  $\rho$ -rule  $\overline{R_1}$  to  $\overline{t}$  using the substitution  $\overline{\sigma_1} = \{F/ z_1 \rightarrow z_2 \rightarrow s(z_2), N/ 0, A/ s(s(0))\}$  and we obtain:

$$\begin{aligned} & \overline{\sigma_1}(F(N, \text{rec}(N, A, x \rightarrow (y \rightarrow (F \ x \ y))))) \\ &= (\overline{\sigma_1}(F)) (\overline{\sigma_1}(N), \text{rec}(\overline{\sigma_1}(N), \overline{\sigma_1}(A), x \rightarrow (y \rightarrow ((\overline{\sigma_1}(F)) \ x \ y)))) \\ &= (z_1 \rightarrow z_2 \rightarrow s(z_2)) (0, \text{rec}(0, s(s(0)), x \rightarrow (y \rightarrow ((z_1 \rightarrow z_2 \rightarrow s(z_2)) \ x \ y)))) \\ &\mapsto_{\rho\sigma} (z_1 \rightarrow z_2 \rightarrow s(z_2)) (0, \text{rec}(0, s(s(0)), x \rightarrow (y \rightarrow s(y)))) \\ &\mapsto_{\rho\sigma} s(\text{rec}(0, s(s(0)), x \rightarrow (y \rightarrow s(y)))) \end{aligned}$$

Proceeding in the reduction of the  $\rho$ -term obtained as intermediary result:

$$\begin{aligned}
& (s(x) \rightarrow s(\overline{R_0} x)) \ s(rec(0, s(s(0)), x \rightarrow (y \rightarrow s(y)))) \\
& \mapsto_\rho [s(x) \leftarrow_{\mathbb{N}} s(rec(0, s(s(0)), x \rightarrow (y \rightarrow s(y))))].s(\overline{R_0} x) \\
& \mapsto_\sigma s(\overline{R_0} (rec(0, s(s(0)), x \rightarrow (y \rightarrow s(y))))
\end{aligned}$$

Finally we apply  $\overline{R_0}$  with the substitution  $\overline{\sigma_0} = \{F/ z_1 \rightarrow z_2 \rightarrow s(z_2), A/ s(s(0))\}$ .

$$s(\overline{R_0} (rec(0, s(s(0)), x \rightarrow (y \rightarrow s(y)))) \mapsto_{\rho\overline{\sigma_0}} s(s(s(0)))$$

and we obtain the expected result.

## 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  $\rho Cal$  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  $\rho$ -term.

This  $\rho$ -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  $\rho Cal$  can be applied here for the construction of an appropriate term only from the set of *CRS*-rules and without without any knowledge on the reduction steps. In fact, the use of a recent definition of iterators in  $\rho Cal$  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  $\rho$ -terms need to be inserted to direct the reduction in the  $\rho Cal$ . Rewrite rules are defined at the object level of the  $\rho Cal$  and they are applied explicitly. The reduction is then performed by the three evaluation rules of the  $\rho Cal$ . 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  $\lambda$ -calculus, while in the  $\rho Cal$  the application of a substitution leads to additional explicit reduction steps. For this reason, we generally have a longer reduction scheme in the  $\rho Cal$  than in the *CRS*.

Since we are mainly interested in the expressive power of the  $\rho Cal$  we have proposed in this paper a translation from *CRS* to  $\rho Cal$  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  $\rho Cal$  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  $\rho\text{Cal}$  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  $\rho\text{Cal}_{\mathfrak{q}}$  composing the translation from the *HRS* to the *CRS* as defined in [22] with the translation from *CRS* to  $\rho\text{Cal}$  we have defined in this paper. Some other higher order systems like the Expression Reduction Systems of Khasidashvili [14] (*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  $\rho\text{Cal}$  and the *ERS* or other systems like the Explicit Reduction Systems of Pagano [21].

## References

1. P. Aczel. A general Church-Rosser theorem. Technical report, University of Manchester, 1978.
2. F. Barbanera, M. Fernández, and H. Geuvers. Modularity of Strong Normalisation and Confluence in the Algebraic  $\lambda$ -Cube. *Journal of Functional Programming*, 7(6):613–660, November 1997.
3. H. Barendregt. *Lambda Calculus: its Syntax and Semantics*. North Holland, 1984.
4. Gilles Barthe, Horatiu Cirstea, Claude Kirchner, and Luigi Liquori. Pure Patterns Type Systems. In *Principles of Programming Languages - POPL2003, New Orleans, USA*. ACM, January 2003.
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 Fabio Gadducci and Ugo 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. R. Hindley. Reductions of residuals are finite. *Transactions of the American Mathematical Society*, 240:345–361, juin 1978.

13. J.P. Jouannaud and M. Okada. Abstract Data Type Systems. *Theoretical Computer Science*, 173(2):349–391, 1997.
14. Z.O. Khasidashvili. Expression reduction systems. In *Proceedings of I. Vekua Institute of Applied Mathematics*, volume 36, pages 200–220, 1990.
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. Special issue in honour of Corrado Böhm.
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 Lambda 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 a 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, 1993.
23. D. A. Wolfram. *The Clausal Theory of Types*, volume 21 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1993.