Expressing Combinatory Reduction Systems Derivations in the Rewriting Calculus

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

September, 2006

Abstract. The last few years have seen the development of the *rewriting calculus* (also called rho-calculus or ρ-calculus) that uniformly integrates first-order term rewriting and the λ-calculus. The combination of these two latter formalisms has been already handled either by enriching first-order rewriting with higher-order capabilities, like in the *Combinatory Reduction Systems* (CRS), or by adding to the λ-calculus algebraic features. The various higher-order rewriting systems and the rewriting calculus share similar concepts and have similar applications, and thus, it is important to compare these formalisms to better understand their respective strengths and differences.

We show in this paper that we can express Combinatory Reduction Systems derivations in terms of rewriting calculus derivations. The approach we present is based on a translation of each possible CRS-reduction into a corresponding ρ-reduction. Since for this purpose we need to make precise the matching used when evaluating CRS, the second contribution of the paper is to present an original matching algorithm for CRS terms that uses a simple term translation and the classical matching of lambda terms.

Keywords: rewriting calculus, Combinatory Reduction Systems, matching

1. Introduction

Lambda calculus and term rewriting provide two fundamental computational paradigms that had a deep influence on the development of proof environments and of programming languages. 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 algebraic features to the λ-calculus. In the first case, we find the works on CRSs (Klop et al., 1993) and other higher-order rewriting systems (Wolfram, 1993; Nipkow and Prehofer, 1998), in the second case the works on the combination of the λ-calculus with term rewriting (Barbanera et al., 1997; Blanqui, 2001b; Blanqui, 2001a; Jouannaud and Okada, 1997) to mention only a few.

The *Combinatory Reduction Systems* (CRSs), introduced by J.-W. Klop (Klop, 1980) are an extension of first-order rewrite systems with a mechanism for bounding variables as 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 the λ-calculus and higher-order matching (*i.e.* matching modulo the β-rule).

In the same line, the rewriting calculus (also called ρ-calculus) (Cirstea and Kirchner, 2001; Cirstea et al., 2001a; Cirstea et al., 2002), extends first-order term rewriting and the λ-calculus. Its main design concept is to make all the basic ingredients of rewriting such as the notions of rule *formation*, *application* and *result* explicit objects. By making the rule application explicit, the calculus emphasizes on the one hand the fundamental role of matching and on the other hand the intrinsic higher-order nature of rewriting. The ρ-calculus offers a broad spectrum of applications due to the two fundamental parameters of the calculus: the equivalence 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 conveniently describe in a uniform but still appropriately tuned manner different calculi. Typical examples are the λ-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 is important to compare these formalisms in order to better understand their respective strengths and differences. Such a comparison between different higher-order formalisms has already been conducted between *Combinatory Reduction Systems* and *Higher-order Rewrite Systems* in (van Oostrom and van Raamsdonk, 1993).

This paper is concerned with the analysis of the relation between the derivations performed respectively in the rewriting calculus and in higher-order rewriting. The evaluation mechanism of *Combinatory Reduction Systems* relies on (higher-order) matching but the corresponding algorithms have not been specified explicitly until now. We first give a precise definition and algorithm for matching CRS-terms and then we give a method for building rewriting calculus terms encoding derivations in *Combinatory Reduction Systems*.

The definition of matching in CRS is done with respect to matching in the λ-calculus, *i.e.* modulo $\beta\eta$. We present the transformations to and from the λ-calculus which allow us to correctly and completely define matching in CRS exactly as the matching of the corresponding λ-terms.

We define then a translation of the various CRS concepts, like terms, assignments, *etc.* to the corresponding notions of ρ-calculus, and, using this translation, propose a method that associates to every derivation of a CRS-term a term with a corresponding derivation in ρ-calculus. More precisely, the corresponding ρ-term is built starting from the rewrite steps performed in the CRS-derivation. Our analysis and comparison between ρ-calculus 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 paper is organized as follows: in Section 2 we briefly present the ρ-calculus through its components. Section 3 provides a description of CRSs and some examples. In Section 4 we show how matching in CRSs can be defined and related to matching in the λ-calculus. In Section 5 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 6 concludes the paper and gives some perspectives to this work.

2. The Rewriting Calculus

We briefly present in what follows the syntax and the semantics of the ρ-calculus. For a more detailed presentation the reader can refer to (Cirstea et al., 2001a; Cirstea et al., 2002) and for a typed version of the calculus to (Barthe et al., 2003; Cirstea et al., 2001b).

2.1. Syntax

In this paper, the symbols $t, u, \ldots$ range over the set $\mathcal{T}$ of terms, the symbols $X, Y, Z, \ldots$ and $x, y, z \ldots$ 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:

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

The set of patterns $\mathcal{P} \subseteq \mathcal{T}$ is a parameter of the calculus and in the most general case it could be as large as the set of all terms $\mathcal{T}$. In many cases we will restrict it to specific classes of terms, as exemplified at the end of this section.

We assume that the (hidden) application operator $(\_\ \_)$ associates 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 $\_ \wr \_$.

To recover the standard algebraic notation, terms of the form $(t_0\, t_1 \cdots t_n)$ are often denoted by $t_0(t_1, \ldots, t_n)$.

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 *delayed matching constraint* $[t_1 \ll t_2]t_3$ which represents a constrained term t_3 where the matching constraint $t_1 \ll t_2$ 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 the substitution σ is the solution of the matching between t_1 and t_2 (see Definition 5) then the delayed matching constraint evaluates to $\sigma(t_3)$. Terms can be grouped together into *structures* built using the symbol "$\wr$". Finally, to emphasize the relationship with term rewriting, an abstraction $t_1 \rightarrowtriangle t_3$ is also called a *ρ-rule*.

A term may be viewed as a *finite labeled tree*, the leaves of which are labeled with variables or constants and the internal nodes of which are labeled with symbols of positive arity. Typically, ρ-terms can be considered as the finite labeled trees build over the signature $\mathcal{F}^{all} = \mathcal{F} \cup \{\_\ \_, \_[\_ \ll \_], \_ \rightarrowtriangle \_, \_ \wr \_\}$ and the set of variables $\mathcal{X}$. The notion of *position* of a term can be intuitively seen as a sequence of naturals describing the path from the root of the term (seen as a tree) to the root of the sub-term at that position. Because it will be used extensively to define and study the translations between formalisms, we recall in details the notion of position.

DEFINITION 1 (Positions). *Let $\mathbb{N}_+$ be the set of positive naturals and $\mathbb{N}_+^*$ the corresponding monoid with neutral element ϵ and the concatenation operator ".". For all $p, q \in \mathbb{N}_+^*$, p is a prefix of q, denoted by $p \leq q$, if it exists $q' \in \mathbb{N}_+^*$ such that $q = p.q'$. The natural number p is a strict prefix of q, denoted by $p < q$, if $p \leq q$ and $p \neq q$.*

A term (seen as a tree) built on a set $\mathcal{C}$ of constructors and of variables $\mathcal{X}$ is an application t from a non-empty part $\mathcal{P}os(t)$ of $\mathbb{N}_+^*$ to $\mathcal{C} \cup \mathcal{X}$ such that $\mathcal{P}os(t)$ is closed under prefix (i.e. if $\omega \in \mathcal{P}os(t)$, then all prefixes of ω belong to $\mathcal{P}os(t)$) and for all $p \in \mathcal{P}os(t)$ and any $i \in \mathbb{N}_+$, $p.i \in \mathcal{P}os(t)$ if and only if $t(p) = c \in \mathcal{C}$ and $1 \leq i \leq arity(c)$ (where $arity(c)$ represents the number of arguments of c).

$\mathcal{P}os(t)$ is called the set of positions (or occurrences) of t, and if t is finite then $\mathcal{P}os(t)$ is finite as well. We call $\epsilon \in \mathcal{P}os(t)$ the empty sequence denoting the head position of t. We denote by $t(\omega)$ the symbol at position ω in t; $t(\epsilon)$ is also called the head symbol of t. A sub-term of t at position $\omega \in \mathcal{P}os(t)$ is denoted by $t_{|\omega}$ and defined by $\forall(\omega.\omega') \in \mathcal{P}os(t), \omega' \in \mathcal{P}os(t_{|\omega}), t_{|\omega}(\omega') = t(\omega.\omega')$. We use the notation $t_{\lceil u \rceil_\omega}$ to state that t has a sub-term u at position ω and the notation $t\lceil \omega \hookleftarrow u \rceil$ to signify that the sub-term $t_{|\omega}$ has been replaced by u in t. To simplify notations we write $t(\omega.i^n.\omega')$ for $t(\omega.\underbrace{(i \ldots i)}_{n}.\omega')$.

Let us emphasize the status of variables in ρ-terms: 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. Formally:

DEFINITION 2 (Free, bound and active variables). *The set of* free *variables of a term is defined as:*

$$\begin{array}{llll}
\mathcal{FV}(f) \triangleq \{\,\} & & \mathcal{FV}(t_1\ t_2) & \triangleq \mathcal{FV}(t_1) \cup \mathcal{FV}(t_2) \\
\mathcal{FV}(X) \triangleq \{X\} & & \mathcal{FV}(t_1 \wr t_2) & \triangleq \mathcal{FV}(t_1) \cup \mathcal{FV}(t_2) \\
\mathcal{FV}(t_1 \rightarrowtriangle t_2) & \triangleq \mathcal{FV}(t_2) \setminus \mathcal{FV}(t_1) & & \\
\mathcal{FV}(t_3[t_1 \ll t_2]) & \triangleq (\mathcal{FV}(t_3) \setminus \mathcal{FV}(t_1)) \cup \mathcal{FV}(t_2) & &
\end{array}$$

The set $\mathcal{BV}$ of bound *variables of a term is the complement of the set of free variables w.r.t. the set of variables of the respective term. A term is called* closed *or* ground *if all its variables are bound. A variable is* active *in a term t if it is free in t and if it appears in the left-hand side of an application occurring in t.*

As in any calculus involving binders, we work modulo the *α-convention* of Church (Church, 1941), *i.e.* two terms that differ only by the names of their bound variables are considered

α-equivalent, and modulo the *hygiene-convention* of Barendregt (Barendregt, 1984), *i.e.* free and bound variables have different names.

An important class of terms consists of ρ-patterns whose definition is directly inspired from the now classical notions of CRS-patterns (Klop, 1980) and λ-patterns. It has the fundamental property that $\beta\eta$-unification of λ-patterns (also called higher-order patterns) is decidable and $\beta\eta$-unifiable λ-patterns have a most general $\beta\eta$-unifier (Miller, 1991).

DEFINITION 3 (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 $Z\ x_1 \ldots\ x_n$ where the variables $x_1, \ldots, x_n$, $n \geq 0$, are distinct and all bound in p.*

For example, $x \to f(Z\ x)$ is a ρ-pattern while $(Z\ x\ x) \to x$ and $g(Z\ x)$ are not since x appears twice on the left-hand side of the abstraction and x is not bound in the latter term.

2.2. Substitutions and matching

The classical notion of simultaneous substitution application used in higher-order calculi, like the λ-calculus, can be straightforwardly adapted to deal with the new forms of constrained terms introduced in ρ-calculus. Notice that if we restrict Definition 4 to terms containing only variables as patterns and no delayed matching constraints or structures then the classical substitution application of the λ-calculus is obtained.

DEFINITION 4 (Substitutions). *A substitution σ is a mapping from the set of variables to the set of terms. A finite substitution has the form $\sigma = \{X_1/t_1 \ldots X_m/t_m\}$ where $Dom(\sigma) = \{X_1, \ldots, X_m\}$. The application of a substitution σ to a term t, denoted by $\sigma(t)$ or $t\sigma$, is defined as follows:*

$$\begin{array}{lll@{\qquad}lll}
\sigma(f) & \triangleq & f & \sigma(t_1 \to t_2) & \triangleq & \sigma(t_1) \to \sigma(t_2) \\
\sigma(X_i) & \triangleq & \begin{cases} t_i & \textit{if } X_i \in Dom(\sigma) \\ X_i & \textit{otherwise} \end{cases} & \begin{array}{l}\sigma(t_3[t_1 \ll t_2]) \\ \sigma(t_1 \wr t_2)\end{array} & \begin{array}{l}\triangleq \\ \triangleq\end{array} & \begin{array}{l}\sigma(t_3)[\sigma(t_1) \ll \sigma(t_2)] \\ \sigma(t_1) \wr \sigma(t_2)\end{array} \\
\sigma(t_1\, t_2) & \triangleq & \sigma(t_1)\ \sigma(t_2) & & &
\end{array}$$

This defines higher-order substitutions as we work modulo the *α-convention*; when applying a substitution to an abstraction, we assume that the free variables of the corresponding abstracted pattern do not belong to the substitution. One can notice that under this assumption $\sigma(t_1 \to t_2) = t_1 \to \sigma(t_2)$ and $\sigma(t_3[t_1 \ll t_2]) = \sigma(t_3)[t_1 \ll \sigma(t_2)]$. When implicit renaming is not assumed a more complex definition should be used, or alternatively, explicit substitutions can be considered (Cirstea et al., 2004a).

The evaluation mechanism of the calculus relies on the fundamental operation of *matching* that allows us to bind variables to their current values. Since we want to define an expressive and powerful calculus, we allow the matching to be performed *modulo* a congruence on terms. This congruence can be defined either equationally when some operators of the signature are associative and commutative or by other means like, for example, considering the $\beta\eta$-equivalence restricted to higher-order patterns *a la* Miller. Since, one of the most usual and convenient approaches consists in introducing the congruence as a theory defined by a set of axioms, we sometimes call the congruence modulo which matching is performed "matching theory".

This congruence used at matching time is a fundamental parameter of the calculus and different instances are obtained when instantiating this parameter by a congruence defined, for example, syntactically, or equationally or in a more elaborated way (Cirstea et al., 2001a). We assume given a congruence $=_{\mathbb{T}}$ and we now recall the definition of matching problems in this general setting.

DEFINITION 5 (Matching modulo $\mathbb{T}$). *Given a congruence $=_{\mathbb{T}}$ on the terms, a* $\mathbb{T}$-matching-equation *is a formula of the form $t_1 \prec\!\!\prec_{\mathbb{T}} t_2$, where t_1 and t_2 are two terms. A substitution σ*

$$
\begin{array}{llll}
(\rho) & (t_1 \rightarrowtriangle t_2)t_3 & \to_{\rho} & t_2[t_1 \ll t_3] \\
(\sigma) & t_2[t_1 \ll t_3] & \to_{\sigma} & \sigma_1(t_2) \wr \ldots \wr \sigma_n(t_2) \wr \ldots \\
 & & & \textit{where } \sigma_i \in \mathit{Sol}(t_1 \prec\!\!\prec_{\mathbb{T}} t_3) \\
(\delta) & (t_1 \wr t_2)\, t_3 & \to_{\delta} & t_1\, t_3 \wr t_2\, t_3
\end{array}
$$

Figure 1. Small-step reduction semantics

is a solution of the matching-equation $t_1 \prec\!\!\prec_{\mathbb{T}} t_2$ *if* $\sigma(t_1) =_{\mathbb{T}} t_2$. *A* $\mathbb{T}$ matching-system *is a conjunction of* $\mathbb{T}$ *matching-equations. A substitution is a solution of a* $\mathbb{T}$ *matching-system* E *if it is a solution of all the* $\mathbb{T}$*-matching-equations in* E*. We denote by Sol(E) the set of all the solutions of* E*.*

If the underlying congruence $=_{\mathbb{T}}$ is clear from the context (usually when matching is performed syntactically) the notation $t_1 \prec\!\!\prec t_2$ is used instead of $t_1 \prec\!\!\prec_{\mathbb{T}} t_2$.

2.3. Semantics

The small-step reduction semantics of the ρ-calculus 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 $\mathbb{T}$) the matching equation $t_1 \prec\!\!\prec_{\mathbb{T}} t_3$, 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 "$\wr$" constructor.

Since the evaluation of a delayed matching constraint depends on the underlying congruence, the delayed matching constraints should be of the form $t_2[t_1{\ll_{\mathbb{T}}}t_3]$. Nevertheless, for the sake of simplicity we will make explicit the congruence only when not clear from the context. According to the specified congruence, the application of the (σ)-rule may produce a structure with a finite or infinite number of elements depending on the number of solutions for the corresponding matching problem. In the following we will restrict to congruences leading to a finite and even unitary set of substitutions and thus, the result of reducing a delayed matching constraint is always a singleton. Consequently, the ρ-terms obtained in Section 5 by translating various CRS-(meta)terms contain no "$\wr$" and their corresponding evaluations do not use the δ rule. Nevertheless, as we briefly discuss in the last section of the paper, the encoding of CRS-evaluations guided by different strategies would need the use of terms built using this constructor..

As usual, we introduce the classical notions of one-step reduction, many-step reduction, and congruence with respect to the relation $\to_{\rho\sigma\delta}$ induced by the top-level rules of the ρ-calculus. The one-step evaluation $\mapsto_{\rho\sigma\delta}$ is the contextual closure of $\to_{\rho\sigma\delta}$. The many-step evaluation $\mapsto\!\!\!\to_{\rho\sigma\delta}$ is defined as the reflexive and transitive closure of $\mapsto_{\rho\sigma\delta}$. The congruence $=_{\rho\sigma\delta}$ is the symmetric and transitive closure of $\mapsto\!\!\!\to_{\rho\sigma\delta}$.

EXAMPLE 6 (Small-step reductions). *We consider the two* ρ*-terms* $(f(x) \rightarrowtriangle (3 \rightarrowtriangle 3)\ x)\ f(3)$ *and* $(f(x) \rightarrowtriangle (3 \rightarrowtriangle 3)\ x)\ f(4)$ *and we show one possible reduction for each of them (corresponding redexes are underlined):*

1. $(f(x) \rightarrowtriangle \underline{(3 \rightarrowtriangle 3)\ x})\ f(3) \mapsto_{\rho} \underline{(f(x) \rightarrowtriangle 3[3 \ll x])\ f(3)} \mapsto_{\rho} \underline{3[3 \ll x][f(x) \ll f(3)]}$
$\mapsto_{\sigma} \underline{3[3 \ll 3]} \mapsto_{\sigma} 3$

2. $\underline{(f(x) \rightarrowtriangle (3 \rightarrowtriangle 3)\ x)\ f(4)} \mapsto_{\rho} \underline{((3 \rightarrowtriangle 3)\ x)[f(x) \ll f(4)]} \mapsto_{\sigma} \underline{(3 \rightarrowtriangle 3)\ 4} \mapsto_{\rho} 3[3 \ll 4]$

2.4. Two useful instances of the rewriting calculus

The general framework of the ρ-calculus can be made specific by instantiating the parameters of the calculus, *i.e.* by choosing the concrete subset of ρ-terms that can be used as patterns and the congruence on terms modulo which the matching is performed. We present below two such instances which will be used later on in the paper.

2.4.1. ρCal_λ and λ-calculus

The ρCal_λ is the instance of ρ-calculus where:

- $\mathcal{P} \triangleq \mathcal{X}$ and,
- the congruence on terms modulo which the matching is performed is the syntactic equality.

We obtain an instance where the set of patterns coincide with the set of variables, *i.e.* all the ρ-rules are of the form $\mathcal{X} \twoheadrightarrow \mathcal{T}$, and where a syntactic matching is used. We should point out that since the matching is unitary, all structures obtained as the result of the application of a rewrite rule contain only one element and, as detailed in (Cirstea and Kirchner, 2001), the reductions in ρCal_λ and λ-calculus are very similar. More precisely, any λ-term can be trivially translated to a ρ-term (only the notation for the abstraction is modified) and each β-reduction step corresponds to a (ρ)-step followed by a (σ)-step. Note also that all the notions defined for the ρ-calculus like free or active variables specialize in a natural way to ρCal_λ.

We denote the reduction relation of this calculus by $\mapsto_{\rho_\lambda}$, its reflexive and transitive closure by $\mapsto\!\!\!\!\to_{\rho_\lambda}$. The generated congruence relation is denoted by $=_{\rho_\lambda}$.

2.4.2. $\rho Cal_{\P}$

Since one of our goals is to define an instance of ρ-calculus with an evaluation mechanism similar to that of CRS, we now introduce an instance of the rewriting calculus which considers higher-order matching on a more general set of patterns. The calculus denoted by $\rho Cal_{\P}$ is defined as the ρ-calculus where:

- $\mathcal{P}$ is the set of ρ-patterns and,
- the congruence on terms modulo which the matching is performed is $=_{\rho_\lambda}$.

When restricting the left-hand sides of matching-equations to ρ-patterns, matching modulo $=_{\rho_\lambda}$ is denoted by $\prec\!\!\prec_{\P}$.

If we assume that all constants and variables are given a fixed arity and that all the terms are considered to be arity compliant (*i.e.* such that any symbol gets all its arguments) then we can show, via a translation into the lambda-calculus similar to the one used in the Section 4, that ρ-pattern matching is decidable and unitary. Consequently, under the above assumptions, the matching involved in the application of the evaluation rule (σ) of $\rho Cal_{\P}$ yields a single substitution.

3. Combinatory Reduction Systems

The *Combinatory Reduction Systems* (CRSs), introduced by J.W. Klop in 1980 (Klop, 1980), are a generalization of first-order term rewrite systems with a mechanism of bound variables like in the λ-calculus. The definitions of this section are based on the presentation of CRSs given in (Klop et al., 1993).

3.1. Terms, metaterms and substitutions

In what follows the symbols $A, B, \dots L, P, R, \dots$ range over the set $\mathcal{MT}$ of so called *metaterms*, the symbols $t, u, \dots$ range over the set $\mathcal{T}_{\text{CRS}}$ 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 symbols $f, g, \dots$ range over the set $\mathcal{F}$ of *functional symbols* of fixed arity. We denote by $\mathcal{F}_i \subset \mathcal{F}$ (respectively $\mathcal{Z}_i \subset \mathcal{Z}$) the subset of symbols (respectively metavariables) of arity i. All symbols can be indexed. We denote by $\equiv$ the syntactic identity of metaterms or substitutions. The set of CRS-metaterms is defined as follows:

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

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

Comparing to first-order rewrite systems, in the syntax of CRSs we have two new concepts: the symbol $[\_]\_$ and the metavariables. The operator $[\_]\_$ denotes an abstraction similar to the abstraction of the λ-calculus such that in $[x]t$ the variable x is bound in t. Metavariables (in the CRS rewrite rules defined below) behave as (free) variables of first-order rewrite systems. Metavariables cannot be bound by the abstraction operator but variables appearing as arguments of a metavariable can be bound by this operator. For example, the only bound variable in $[x]Z(x)$ is x.

In a metaterm of the form $[x]t$ we call t the scope of $[x]$. A variable x occurs *free* in a metaterm if it is not in the scope of an occurrence of $[x]$. A variable x occurs *bound* otherwise. The set of free variables of a metaterm A is written as $\mathcal{FV}(A)$. The set of metavariables of a metaterm A is written as $\mathcal{MV}(A)$.

As for the ρ-calculus, we work modulo the α-conversion and Barendregt's *hygiene-convention.*

EXAMPLE 7 (Terms and Metaterms). *Some examples of terms and metaterms:*

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

The application of substitutions to metavariables is defined at the meta-level of the calculus and uses the $\underline{\lambda}$-calculus as meta-language (underlined just for distinguishing it from classical λ-calculus). Unintended bindings of variables by the $\underline{\lambda}$-abstractor operator are avoided using α-conversion. To simplify the notation we denote $\underline{\lambda}x_1 \dots \underline{\lambda}x_n.t$ by $\underline{\lambda}x_1 \dots x_n.t$. The reduction of $\underline{\lambda}$-redexes is performed by the $\underline{\beta}$-rule of the $\underline{\lambda}$-calculus. The relation "t_1 β-reduces in one step to t_2" is denoted by $t_1 \mapsto_\beta t_2$ and is defined as the context closure of the relation generated by the β-rule. The relation $t_1 \mapsto\!\!\!\rightarrow_\beta t_2$ (t_1 β-reduces to t_2) is defined as the reflexive and transitive closure of the relation $\mapsto_\beta$. The relation $t_1 =_\beta t_2$ (t_1 and t_2 are β-equivalent) is defined as the symmetric and transitive closure of $\mapsto\!\!\!\rightarrow_\beta$. We use the same (underlined) notations for the underlined relation. We denote by $t{\downarrow_\beta}$ the β-normal form of a term t (when it exists) and by $t{\downarrow_{\underline{\beta}}}$ the $\underline{\beta}$-normal form of the term t. We should point out that a CRS-(meta)term is necessarily in $\underline{\beta}$-normal form.

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

DEFINITION 8 (Substitute). *An n-ary* substitute *is an expression of the form* $\xi = \underline{\lambda}x_1 \dots x_n.u$ *(sometimes denoted by* $\underline{\lambda}\overline{x}.u$*) where* $x_1, \dots, x_n$ *are distinct variables and* u *is a* CRS*-term and its application to an n-tuple of* CRS*-terms* $(t_1, \dots, t_n)$ *yields the simultaneous substitution of* $x_1, \dots, x_n$ *by* $t_1, \dots, t_n$ *in* u*, denoted by:* $(\underline{\lambda}x_1 \dots x_n.u)(t_1, \dots, t_n){\downarrow_{\underline{\beta}}} = u\{x_1/t_1, \dots, x_n/t_n\}$.

DEFINITION 9 (Assignment). *An* assignment $\sigma = \{(Z_1, \xi_1), \ldots, (Z_n, \xi_n)\}$, *is a finite set of pairs (metavariable, substitute) such that arity(Z_i) = arity(ξ_i) $\forall i \in \{1, \ldots, n\}$ (Dom(σ) = $\{Z_1, \ldots, Z_n\}$). The application of an assignment σ to a* CRS*-metaterm t, denoted by $\sigma(t)$ or σt, is inductively defined by:*

$$
\begin{array}{llll}
\sigma(x) \triangleq x & & \sigma([x]t) & \triangleq [x]\sigma(t) \\
\sigma(Z_i) \triangleq \xi_i & \text{if } (Z_i, \xi_i) \in \sigma & \sigma(f(t_1, \ldots, t_m) & \triangleq f(\sigma(t_1), \ldots, \sigma(t_m)) \\
\sigma(Z_i) \triangleq Z_i & \text{if } Z_i \notin \mathit{Dom}(\sigma) & \sigma(Z_i(t_1, \ldots, t_m)) & \triangleq \sigma(Z_i)(\sigma(t_1), \ldots, \sigma(t_m))\downarrow_{\underline{\beta}}
\end{array}
$$

Notice that the assignments have no effect on variables since they can instantiate only metavariables. Since we work modulo the α-conversion, unintended bindings of free variables are avoided by renaming bound variables; for example, if $\sigma(Z) = x$ then we can suppose that the variable y in $[y]Z$ is always different from x and thus $\sigma([y]Z) = [y]\sigma(Z)$.

The instantiation of a term is defined by replacing each metavariable by a substitute (*i.e.* a $\underline{\lambda}$-term) and by reducing all residuals of $\underline{\beta}$-redexes that are present in the initial term, *i.e.* performing a so called *development* (Barendregt, 1984) on $\underline{\lambda}$-terms. Since it is well known that in the 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 CRS-term.

3.2. Rules and rewriting

A CRS rewrite rule is a pair of metaterms. Its metavariables define the reduction schemes since they can be instantiated by any term. We consider as left-hand side of the rules only the CRS-metaterms satisfying the CRS-pattern definition:

DEFINITION 10 (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.*

In his thesis (Klop, 1980), J. W. Klop also considers general terms as left-hand sides of rewrite rules but the corresponding matching becomes much more elaborated in this case.

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

DEFINITION 11 (Rewrite rules). *A set of* CRS *rewrite rules consists of rules of the form $L \rightarrow R$ satisfying the following conditions:*

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

The first three conditions are the usual ones used in first-order rewriting: the first condition states that rules are built from metaterms; the second one specifies the structure of left-hand sides; the third one avoids the introduction of arbitrary terms. As we will see next, the last condition ensures the decidability and the uniqueness of the solution of the matching inherent to the application of the CRS-rules.

EXAMPLE 12 (β-rule in CRSs). *The β-rule of the λ-calculus $(\lambda x.t)u \rightarrow_\beta t\{x/u\}$ can be expressed as the following rewrite rule:*

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

In the above rule, called BetaCRS *in this paper, $\mathit{App} \in \mathcal{F}_2$ and $\mathit{Ab} \in \mathcal{F}_1$ are the encodings for the λ-calculus application and abstraction operators respectively.*

$$\begin{tikzcd}
L \prec\!\!\prec_{\mathrm{CRS}} t \arrow[r, bend left=10, "Solved\ by"] \arrow[d, "\Pi_\lambda"'] & \Pi_{\mathrm{CRS}}(\sigma_\lambda) = \{(Z_1, \Pi_{\mathrm{CRS}}(\xi_1)), \ldots, (Z_n, \Pi_{\mathrm{CRS}}(\xi_n))\} \\
\Pi_\lambda(L) \prec\!\!\prec_{\beta\eta} \Pi_\lambda(t) \arrow[r, "Solving"'] & \sigma_\lambda = \{(Z_1/\xi_1), \ldots, (Z_n/\xi_n)\} \arrow[u, "\Pi_{\mathrm{CRS}}"']
\end{tikzcd}$$

Figure 2. Translation of CRS matching problems

3.3. Semantics

Given a rewrite rule $L \to R$ and a substitution σ, we have $\sigma(L) \to_{L \to R} \sigma(R)$ if $\sigma(L), \sigma(R) \in \mathcal{T}_{\mathrm{CRS}}$. 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 by $\mapsto_{L \to R}$ if we want to specify the applied rule) is the context closure of the relation induced (as above) by the rules in $\mathcal{R}$. The many-step evaluation $\mapsto\!\!\!\!\to_{\mathcal{R}}$ is defined as the reflexive and transitive closure of $\mapsto_{\mathcal{R}}$.

EXAMPLE 13. *Let us consider the* CRS*-term* $f(App(Ab([x]f(x)), a))$. *We apply to the sub-term* $App(Ab([x]f(x)), a)$ *the* BetaCRS *rule (Example 12). A solution of the corresponding matching problem is the assignment* $\sigma = \{(Z, \underline{\lambda}y.fy), (Z_1, a)\}$, *since, when applying it to L, we obtain precisely* $App(Ab([x]f(x)), a)$*:*

$$\begin{aligned} \sigma(L) &= \sigma(App(Ab([x]Z(x)), Z_1)) = App(Ab([x]\sigma(Z)(x), \sigma(Z_1))\downarrow_{\underline{\beta}} = App(Ab([x](\underline{\lambda}y.fy)(x), a))\downarrow_{\underline{\beta}} \\ &= App(Ab([x]f(x)), a) \end{aligned}$$

As the result of the rule application, we obtain the instantiation by σ *of the right-hand side* R *of the rule* BetaCRS*:* $\sigma(R) = \sigma(Z(Z_1)) = (\sigma(Z))(\sigma(Z_1))\downarrow_{\underline{\beta}} = (\underline{\lambda}y.fy)(a)\downarrow_{\underline{\beta}} = f(a)$. *Therefore we have* $App(Ab([x]f(x)), a) \mapsto_{\mathrm{BetaCRS}} f(a)$ *and thus* $f(App(Ab([x]f(x)), a)) \mapsto_{\mathrm{BetaCRS}} f(f(a))$.

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

4. CRS Matching

As we have seen in the previous section, a CRS-rule can be applied to a CRS-term t if there exists an assignment whose application to the left-hand side of the rule leads to the term t. The mechanism used to determine this assignment is called, as usually, matching. CRSs use higher-order matching, but the way the assignment is algorithmically obtained has not been specified explicitly in previous works on CRS. We describe in what follows how a solution of a CRS matching problem can be exhibited by translating the problem into a higher-order matching problem on λ-terms and translating back the obtained substitution.

The approach we propose here is summarized by the translation schema depicted in Figure 2. The projection function Π_λ translates CRS-metaterms into simply typed λ-terms and thus CRS matching-equations (denoted using $\prec\!\!\prec_{\mathrm{CRS}}$) into higher-order matching-equations (denoted using $\prec\!\!\prec_{\beta\eta}$) on λ-terms. We can then use one of the already known algorithms for higher-order pattern matching to find the substitution σ_λ which solves this latter matching-equation. Finally, the (back) projection function Π_{CRS} translates λ-terms into CRS-terms and thus the obtained substitution σ_λ into a well-defined CRS-assignment.

In what follows we recall the definitions of higher-order matching and CRS matching, we define the translation functions and we prove the correctness of our approach.

4.1. Higher-order and CRS matchings

The λ-calculus provides a semantical framework for many programming languages like λProlog (Miller and Nadathur, 1986) or ML. In these cases, the notion of equality between λ-terms is typically defined modulo α-conversion and $\beta\eta$-reduction. Unification, typically needed for theorem proving or for logic programming, is extended to higher-order unification (Huet, 1975) which is undecidable in general (Goldfarb, 1981). Therefore the class of considered terms is often limited to λ-patterns (simply typed λ-terms in long $\beta\eta$-normal form where the arguments of a free variable are always η-equal to distinct bound variables), for which the unification (and matching) problem is decidable, unitary (Miller, 1991) and even linear (Qian, 1993).

DEFINITION 14 (Higher-order matching). *Let t_1 and t_2 be two λ-terms. A substitution σ is a solution of the* higher-order matching-equation $t_1 \prec\!\!\!\prec_{\beta\eta} t_2$ *if $\sigma(t_1) =_{\beta\eta} t_2$. A* higher-order matching-system *is a conjunction of higher-order matching-equations.*

We can notice that a similar notion of pattern is used in CRS and that matching in CRS can be compared to higher-order matching. Moreover, even if a CRS matching algorithm has not been clearly specified until now, the way the CRS substitution works naturally suggests the use of higher-order matching.

DEFINITION 15 (CRS Matching). *Let A_1 and A_2 be two* CRS*-metaterms. An assignment σ is a solution of the* CRS matching-equation $A_1 \prec\!\!\!\prec_{\text{CRS}} A_2$ *if $\sigma(A_1) \equiv A_2$. A* CRS matching-system *is a conjunction of* CRS *matching-equations.*

Since matching is mainly used to check if a CRS rewrite rule can be applied to a CRS-term, we restrict in the following to matching-equations having a CRS-pattern on the left-hand side and a CRS-term on the right-hand side.

We should point out that we use the same notation $\sigma(\_)$ for both λ-substitution application and CRS-assignment application, but in the case of an assignment some "hidden" $\underline{\beta}$-steps are possibly included in the application (see Example 13).

4.2. From CRS to the λ-calculus

We define here a translation function of CRS-metaterms into simply typed λ-terms strongly inspired from (van Oostrom and van Raamsdonk, 1993) as well as the function that computes the corresponding positions in the translated terms. The types of the λ-terms are built from only one base type that we denote τ.

DEFINITION 16. *The function Π_λ translates* CRS*-terms into simply typed λ-terms*

- $\Pi_\lambda(x) = x$ *with x of type τ $(x : \tau)$,*
- $\Pi_\lambda(f(t_1, \ldots, t_n)) = f\ \Pi_\lambda(t_1) \ldots \Pi_\lambda(t_n)$ *with $f : \tau \rightarrow \ldots \rightarrow \tau$ (n type arrows),*
- $\Pi_\lambda(Z(t_1, \ldots, t_n)) = Z\ \Pi_\lambda(t_1) \ldots \Pi_\lambda(t_n)$ *with $Z : \tau \rightarrow \ldots \rightarrow \tau$ (n type arrows),*
- $\Pi_\lambda([x]t) = \Lambda\ \lambda x.\Pi_\lambda(t)$ *with $\Lambda : (\tau \rightarrow \tau) \rightarrow \tau$ (collapsing symbol),*

and assignments into substitutions:

- $\Pi_\lambda(\{\ldots, (Z, t), \ldots\}) = \{\ldots, Z/\Pi_\lambda(t), \ldots\}$, *and substitutes into λ-terms:*

- $\Pi_\lambda(\underline{\lambda}\overline{x}.t) = \lambda\overline{x}.\Pi_\lambda(t)$.

Since we are mainly interested in the matching problems induced by the application of CRS rules we concentrate on matching-equations of the form $L \prec\!\!\prec_{\text{CRS}} t$ where $L \in \mathcal{MT}$ is a closed pattern and $t \in \mathcal{T}_{\text{CRS}}$ and we can immediately notice that the λ-terms obtained when translating this kind of CRS-terms have certain properties.

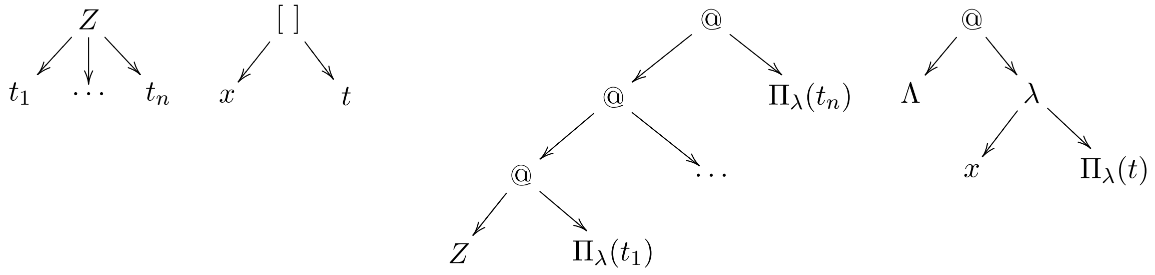
PROPOSITION 17. *For any closed pattern $L \in \mathcal{MT}$ with $\mathcal{MV}(L) = \{Z_1, \dots, Z_n\}$, and any term $t \in \mathcal{T}_{\text{CRS}}$ we have:*

- $\Pi_\lambda(L)$ *and* $\Pi_\lambda(t)$ *are in long βη-normal form;*
- $\Pi_\lambda(L) : \tau$ *and* $\Pi_\lambda(t) : \tau$*;*
- $\Pi_\lambda(L)$ *is a λ-pattern;*
- $\mathcal{FV}(\Pi_\lambda(L)) = \{Z_1, \dots, Z_n\}$*;*

Proof. By structural induction. We use the fact that all CRS metaterms are well-formed and the translation function Π_λ yields no β-redexes (no applications of λ-abstractions to some other terms can be generated). We should also notice that the only applications appearing in the λ-term obtained as translation of a CRS-term (which, by definition, contains no explicit application operator and no metavariables) are of the form $(((f\ u_1)\ u_2)\dots u_n)$, where f is a constant and $u_1, \dots, u_n$ are λ-terms.

Once we have translated the CRS-metaterms appearing on the two sides of a CRS matching-equation, we can compute the solution of the obtained higher-order matching problem using, for example, the algorithm proposed in (Miller, 1991). Since the matching is performed against λ-patterns, the substitution that solves the matching-equation $\Pi_\lambda(L) \prec\!\!\prec_{\beta\eta} \Pi_\lambda(t)$, if it exists, is unique and has the form $\sigma = \{Z_1/\xi_1, \dots, Z_n/\xi_n\}$ where ξ_i, $i \in 1, \dots, n$, are λ-terms in long βη-normal form.

The translation function Π_λ induces a bijective correspondence between the sub-terms of a CRS-term and the sub-terms (of type τ) of its translation. The position ω of a sub-term B in a CRS-term $A_{\lceil B \rceil_\omega}$ and the position of its translation $\Pi_\lambda(B)$ in the λ-term $\Pi_\lambda(A)$ are not the same. In fact, since n-ary metavariables and n-ary functions are translated into the λ-calculus as applications of a variable or a function to n λ-terms, the n-ary tree of the CRS-term becomes a binary tree in the λ-calculus:



CRS-terms λ-terms

We introduce therefore a function π_λ defining the sub-term positions after the translation into the λ-calculus.

DEFINITION 18. *Let A be a CRS-metaterm and $\omega \in \mathcal{P}os(A)$ a position in the metaterm A. The function π_λ is inductively defined by:*

- $\pi_\lambda(\epsilon, A) = \epsilon$,
- $\pi_\lambda((n-i).\omega, Z(A_1, \dots, A_n)) = 1^i.2.\pi_\lambda(\omega, A_{(n-i)})$, *where* $i \in \{0, \dots, n-1\}$,
- $\pi_\lambda((n-i).\omega, f(A_1, \dots, A_n)) = 1^i.2.\pi_\lambda(\omega, A_{(n-i)})$, *where* $i \in \{0, \dots, n-1\}$,
- $\pi_\lambda(1.\epsilon, [x]A) = 2.1.\pi_\lambda(\epsilon, x)$,
- $\pi_\lambda(2.\omega, [x]A) = 2.2.\pi_\lambda(\omega, A)$.

4.3. Back from the λ-calculus to crs

The next step consists in defining a function $\Pi_{\textsc{crs}}$ which translates a substitution obtained as a solution of a higher-order matching problem into CRS-assignments. Since we want to translate only substitutions obtained as the solutions of a higher-order matching problem corresponding to a CRS-matching problem, the translation function $\Pi_{\textsc{crs}}$ is defined only on simply typed λ-terms in long $\beta\eta$-normal form. Since $\Lambda\ u$ has type τ and all constants and variables are of type τ then u has necessarily the form $\lambda x.t$ with t of type τ.

DEFINITION 19. *The function* $\Pi_{\textsc{crs}}$ *translates substitutions into assignments:*

- $\Pi_{\textsc{crs}}(\{\dots, Z/\xi, \dots\}) = \{\dots, (Z, \Pi_{\textsc{crs}}(\xi), \dots\}$,
  with $arity(Z) = n$ *if* $Z : \tau \rightarrowtriangle \dots \rightarrowtriangle \tau$ *(n type arrows),*

and simply typed λ*-terms in long* $\beta\eta$*-normal form into* CRS*-terms*

- $\Pi_{\textsc{crs}}(x) = x$,
- $\Pi_{\textsc{crs}}(Z\ t_1 \dots t_n) = Z(\Pi_{\textsc{crs}}(t_1), \dots, \Pi_{\textsc{crs}}(t_n))$ *if* $Z : \tau \rightarrowtriangle \dots \rightarrowtriangle \tau$ *(n type arrows),*
- $\Pi_{\textsc{crs}}(f\ t_1 \dots t_n) = f(\Pi_{\textsc{crs}}(t_1), \dots, \Pi_{\textsc{crs}}(t_n))$ *if* $f : \tau \rightarrowtriangle \dots \rightarrowtriangle \tau$ *(n type arrows),*
- $\Pi_{\textsc{crs}}(\Lambda\ \lambda x.t) = [x]\Pi_{\textsc{crs}}(t)$,
- $\Pi_{\textsc{crs}}(\lambda\overline{x}.t) = \underline{\lambda}\overline{x}.\Pi_{\textsc{crs}}(t)$.

We can notice that the obtained assignments are well-defined: since ξ is in long $\beta\eta$-normal form and has the same type as Z, ξ has the form $\lambda x_1 \dots x_n.u$ and consequently $\Pi_{\textsc{crs}}(\xi)$ has the form $\underline{\lambda}x_1 \dots x_n.\Pi_{\textsc{crs}}(u)$. Notice also that the function $\Pi_{\textsc{crs}}$ is a partial function that is complete over the image of Π_λ.

As for the previous translation function, we define a function $\pi_{\textsc{crs}}$ which computes the position of a given sub-term after translation.

DEFINITION 20. *The function* $\pi_{\textsc{crs}}$ *taking as arguments a simply typed* λ*-term in long* $\beta\eta$*-normal form and a position (of a* λ*-term in long* $\beta\eta$*-normal form) in this term and returning a position in the* $\Pi_{\textsc{crs}}$*-translation of the* λ*-term is inductively defined by:*

- $\pi_{\textsc{crs}}(\epsilon, u) = \epsilon$,
- $\pi_{\textsc{crs}}(1^i.2.\omega, Z\ u_1 \dots u_n) = (n-i).\pi_{\textsc{crs}}(\omega, u_{(n-i)})$ *where* $i \in \{0, \dots, n-1\}$,
- $\pi_{\textsc{crs}}(1^i.2.\omega, f\ u_1 \dots u_n) = (n-i).\pi_{\textsc{crs}}(\omega, u_{(n-i)})$ *where* $i \in \{0, \dots, n-1\}$,
- $\pi_{\textsc{crs}}(2.1.\epsilon, \Lambda\ \lambda x.t) = 1.\pi_{\textsc{crs}}(\epsilon, x)$,
- $\pi_{\textsc{crs}}(2.2.\omega, \Lambda\ \lambda x.t) = 2.\pi_{\textsc{crs}}(\omega, t)$.

Notice that the positions of the Λ symbol in the λ-term are no longer valid in the corresponding CRS-term where they disappear. As for the previous function, the function π_{CRS} is a partial function but complete over the set of positions obtained as result of applying the translation function π_λ.

4.4. Examples

Let us present some examples of CRS matching problems and solutions with the corresponding translations in the λ-calculus. We will show in Section 4.5 that the approach we propose is correct and complete.

EXAMPLE 21. *Given the* CRS*-metaterm* $L = [x]Z(x)$ *and the* CRS*-term* $t = [x]f(x)$ *we show how the proposed approach can be used to solve the* CRS *matching-equation* $L \prec\!\!\prec_{\mathrm{CRS}} t$. *By translating these two terms into λ-terms (the types are omitted) we obtain* $\Pi_\lambda(L) = \Lambda\ \lambda x.Zx$ *and* $\Pi_\lambda(t) = \Lambda\ \lambda x.fx$. *The solution of the matching-equation* $\Pi_\lambda(L) \prec\!\!\prec_{\beta\eta} \Pi_\lambda(t)$ *is the substitution* $\sigma = \{Z/\lambda y.f\ y\}$ *and when translating it into a* CRS *assignment we obtain* $\Pi_{\mathrm{CRS}}(\sigma) = \{Z/\underline{\lambda}y.\Pi_{\mathrm{CRS}}(f\ y)\} = \{Z/\underline{\lambda}y.f(y)\}$.

We can verify that this last assignment is a solution of the CRS *matching-equation* $L \prec\!\!\prec_{\mathrm{CRS}} t$:

$$\{Z/\underline{\lambda}y.f(y)\}([x]Z(x)) = [x]\{Z/\underline{\lambda}y.f(y)\}(Z)(x) = [x](\underline{\lambda}y.f(y))(x)\downarrow_{\underline{\beta}} = [x]f(x)$$

EXAMPLE 22. *We consider the* CRS*-metaterm* $L = App([x]Z_1(x), Z_2)$ *where* $Z_1 \in \mathcal{Z}_1$ *and* $Z_2 \in \mathcal{Z}_0$ *and the* CRS*-term* $t = App([x][y]f(x,y), a)$ *and we solve the* CRS *matching-equation* $L \prec\!\!\prec_{\mathrm{CRS}} t$. *The translation of these two terms into the λ-calculus are* $\Pi_\lambda(L) = App\ (\Lambda\ \lambda x.Z_1\ x)\ Z_2$ *and* $\Pi_\lambda(t) = App\ (\Lambda\ \lambda x.(\Lambda\ \lambda y.f\ x\ y))\ a$, *and the solution of the higher-order matching-equation* $\Pi_\lambda(L) \prec\!\!\prec_{\beta\eta} \Pi_\lambda(t)$ *is the substitution* $\sigma = \{Z_1/\lambda z.(\Lambda\ \lambda y.f\ z\ y), Z_2/a\}$. *When translating this substitution into an assignment we obtain* $\Pi_{\mathrm{CRS}}(\sigma) = \{(Z_1, \underline{\lambda}z.\Pi_{\mathrm{CRS}}(\Lambda\ \lambda y.f\ z\ y)), (Z_2, a)\} = \{(Z_1, \underline{\lambda}z.[y]\Pi_{\mathrm{CRS}}(f\ z\ y)), (Z_2, a)\} = \{(Z_1, \underline{\lambda}z.[y]f(z,y)), (Z_2, a)\}$.

We can verify that this last assignment is a solution of the CRS *matching-equation* $L \prec\!\!\prec_{\mathrm{CRS}} t$:

$$\begin{aligned}
\Pi_{\mathrm{CRS}}(\sigma)(L) &= \Pi_{\mathrm{CRS}}(\sigma)(App([x]Z_1(x), Z_2)) \\
&= App([x]\Pi_{\mathrm{CRS}}(\sigma)Z_1(x), \Pi_{\mathrm{CRS}}(\sigma)(Z_2)) \\
&= App([x]\{(Z_1, \underline{\lambda}z.[y]f(z,y)), (Z_2, a)\}Z_1(x), \{(Z_1, \underline{\lambda}z.[y]f(z,y)), (Z_2, a)\}(Z_2)) \\
&= App([x](\underline{\lambda}z.[y]f(z,y))(x)\downarrow_{\underline{\beta}}, a\downarrow_{\underline{\beta}}) \\
&= App([x][y]f(x,y), a) = t
\end{aligned}$$

4.5. Properties

We show in what follows the soundness and completeness of the translations, *i.e.* that a substitution is a solution of a CRS matching-equation if and only if its translation is a solution of the corresponding higher-order matching-equation. First of all, we state some auxiliary lemmas concerning essentially the context stability and the duality of the two translation functions Π_λ and Π_{CRS}.

LEMMA 23 (Context stability). *Given a simply typed λ-term* $u_{\lceil v\rceil_\omega} : \tau$ *in long $\beta\eta$-normal form then for all ω such that* $v : \tau$ *we have*

$$\Pi_{\mathrm{CRS}}(u_{\lceil v\rceil_\omega}) = \Pi_{\mathrm{CRS}}(u)_{\lceil \Pi_{\mathrm{CRS}}(v)\rceil_{\pi_{\mathrm{CRS}}(\omega,u)}} \tag{1}$$

Given a CRS*-metaterm* $A_{\lceil B\rceil_\omega}$, *then*

$$\Pi_\lambda(A_{\lceil B\rceil_\omega}) = \Pi_\lambda(A)_{\lceil \Pi_\lambda(B)\rceil_{\pi_\lambda(\omega,A)}} \tag{2}$$

Proof. For the first equality we proceed by structural induction on the λ-term u.
Base case: If $u = x$ or $u = f$ then $\omega = \epsilon$ and the result follows immediately.
Induction: u is an application. The cases where $\omega = \epsilon$ are obvious.

- If $u = (f\ u_1 \dots u_n)$, since ω is the position of a sub-term of type τ, then $\omega = 1^i.2.\omega'$ with ω' a position in $u_{(n-i)}$ and we have:

$$\begin{aligned}
\Pi_{\textsc{crs}}((f\ u_1 \dots u_n)_{\lceil v \rceil_{1^i.2.\omega'}}) &= \Pi_{\textsc{crs}}(f\ u_1 \dots u_{(n-i)\lceil v \rceil_{\omega'}} \dots u_n) \\
&= f(\Pi_{\textsc{crs}}(u_1), \dots, \Pi_{\textsc{crs}}(u_{(n-i)\lceil v \rceil_{\omega'}}), \dots, \Pi_{\textsc{crs}}(u_n)) \\
&\overset{ih}{=} f(\Pi_{\textsc{crs}}(u_1), \dots, \Pi_{\textsc{crs}}(u_{(n-i)})_{\lceil \Pi_{\mathrm{CRS}}(v) \rceil_{\pi_{\mathrm{CRS}}(\omega', u_{(n-i)})}}, \dots, \Pi_{\textsc{crs}}(u_n)) \\
&= f(\Pi_{\textsc{crs}}(u_1), \dots, \Pi_{\textsc{crs}}(u_n))_{\lceil \Pi_{\mathrm{CRS}}(v) \rceil_{(n-i).\pi_{\mathrm{CRS}}(\omega', u_{(n-i)})}} \\
&= \Pi_{\textsc{crs}}(f\ u_1 \dots u_n)_{\lceil \Pi_{\mathrm{CRS}}(v) \rceil_{\pi_{\mathrm{CRS}}(1^i.2.\omega', f\ u_1 \dots u_n)}}
\end{aligned}$$

- If $u = (Z\ u_1 \dots u_n)$ then we proceed as in the previous case.
- If $u = \Lambda\ \lambda x.t$, since $\Lambda : (\tau \rightarrowtriangle \tau) \rightarrowtriangle \tau$, then $\omega = 2.\omega'$ with ω' a (non head) position in $\lambda x.t$ and thus we have $\Pi_{\textsc{crs}}((\Lambda\ \lambda x.t)_{\lceil v \rceil_{2.\omega'}}) = \Pi_{\textsc{crs}}(\Lambda\ (\lambda x.t)_{\lceil v \rceil_{\omega'}})$ with either $\omega' = 1.\epsilon$ and $v = x$ or $\omega' = 2.\omega''$ with ω'' a position in t:

1. $\Pi_{\textsc{crs}}((\Lambda\ \lambda x.t)_{\lceil x \rceil_{2.1.\epsilon}}) = ([x]\Pi_{\textsc{crs}}(t))_{\lceil x \rceil_{1.\epsilon}} = \Pi_{\textsc{crs}}(\Lambda\ \lambda x.t)_{\lceil \Pi_{\mathrm{CRS}}(x) \rceil_{\pi_{\mathrm{CRS}}(2.1.\epsilon, \Lambda\ \lambda x.t)}}$
2. $$\begin{aligned}
\Pi_{\textsc{crs}}((\Lambda\ \lambda x.t)_{\lceil v \rceil_{2.2.\omega''}}) &= \Pi_{\textsc{crs}}(\Lambda\ \lambda x.t_{\lceil v \rceil_{\omega''}}) = [x]\Pi_{\textsc{crs}}(t_{\lceil v \rceil_{\omega''}}) \\
&\overset{ih}{=} [x]\Pi_{\textsc{crs}}(t)_{\lceil \Pi_{\mathrm{CRS}}(v) \rceil_{\pi_{\mathrm{CRS}}(\omega'', t)}} \\
&= ([x]\Pi_{\textsc{crs}}(t))_{\lceil \Pi_{\mathrm{CRS}}(v) \rceil_{2.\pi_{\mathrm{CRS}}(\omega'', t)}} \\
&= \Pi_{\textsc{crs}}(\Lambda\ \lambda x.t)_{\lceil \Pi_{\mathrm{CRS}}(v) \rceil_{\pi_{\mathrm{CRS}}(2.\omega'', \lambda x.t)}} \\
&= \Pi_{\textsc{crs}}(\Lambda\ \lambda x.t)_{\lceil \Pi_{\mathrm{CRS}}(v) \rceil_{\pi_{\mathrm{CRS}}(2.2.\omega'', \Lambda\ \lambda x.t)}}
\end{aligned}$$

For the second equality we proceed by structural induction on the CRS-metaterm A.
Base case: If $A = x$ or $A = f$ then $\omega = \epsilon$ and the result follows immediately.
Induction: The cases where $\omega = \epsilon$ are obvious and for the others we have:

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

$$\begin{aligned}
\Pi_\lambda(f(A_1, \dots, A_n)_{\lceil B \rceil_{(n-i).\omega'}}) &= \Pi_\lambda(f(A_1, \dots, A_{(n-i)\lceil B \rceil_{\omega'}}, \dots, A_n)) \\
&= f\ \Pi_\lambda(A_1) \dots \Pi_\lambda(A_{(n-i)\lceil B \rceil_{\omega'}}) \dots \Pi_\lambda(A_n) \\
&\overset{ih}{=} f\ \Pi_\lambda(A_1) \dots \Pi_\lambda(A_{(n-i)})_{\lceil \Pi_\lambda(B) \rceil_{\pi_\lambda(\omega', A_{(n-i)})}} \dots \Pi_\lambda(A_n) \\
&= (f\ \Pi_\lambda(A_1) \dots \Pi_\lambda(A_n))_{\lceil \Pi_\lambda(B) \rceil_{1^i.2.\pi_\lambda(\omega', A_{(n-i)})}} \\
&= \Pi_\lambda(f(A_1, \dots, A_n))_{\lceil \Pi_\lambda(B) \rceil_{\pi_\lambda((n-i).\omega', f(A_1, \dots, A_n))}}
\end{aligned}$$

- If $A = Z(A_1, \dots, A_n)$ then we proceed as in the previous case.
- If $A = [x]A'$ then ω is either of the form $\omega = 1.\epsilon$ and $v = x$ or of the form $\omega = 2.\omega'$ with ω' a position in A':

1. $\Pi_\lambda(([x]A')_{\lceil x \rceil_{1.\epsilon}}) = (\Lambda\ \lambda x.\Pi_\lambda(A'))_{\lceil x \rceil_{2.1.\epsilon}} = \Pi_\lambda([x]A')_{\lceil \Pi_\lambda(x) \rceil_{\pi_\lambda(1.\epsilon, [x]A')}}$
2. $$\begin{aligned}
\Pi_\lambda(([x]A')_{\lceil B \rceil_{2.\omega'}}) &= \Pi_\lambda([x]A'_{\lceil B \rceil_{\omega'}}) = \Lambda\ \lambda x.\Pi_\lambda(A'_{\lceil B \rceil_{\omega'}}) \\
&\overset{ih}{=} \Lambda\ \lambda x.(\Pi_\lambda(A')_{\lceil \Pi_\lambda(B) \rceil_{\pi_\lambda(\omega', A')}}) \\
&= (\Lambda\ \lambda x.\Pi_\lambda(A'))_{\lceil \Pi_\lambda(B) \rceil_{2.2.\pi_\lambda(\omega', A')}} \\
&= \Pi_\lambda([x]A')_{\lceil \Pi_\lambda(B) \rceil_{\pi_\lambda(2.\omega', [x]A')}}
\end{aligned}$$

LEMMA 24 (Inverse). *Given a simply typed λ-term $u : \tau$ in long $\beta\eta$-normal form then we have*

$$\Pi_\lambda(\Pi_{\text{CRS}}(u)) = u \tag{1}$$

Given a CRS*-metaterm $A \in \mathcal{MT}$, then*

$$\Pi_{\text{CRS}}(\Pi_\lambda(A)) = A \tag{2}$$

Proof. For the first equality we proceed by structural induction on the λ-term u.
Base case: If $u = x$ then $\Pi_\lambda(\Pi_{\text{CRS}}(x)) = \Pi_\lambda(x) = x$. The case $u = f$ is similar.
Induction: The cases where $\omega = \epsilon$ are obvious and for the others we have:.

- If $u = f\ t_1 \dots t_n$ then we obtain $\Pi_\lambda(\Pi_{\text{CRS}}(f\ t_1 \dots t_n)) = \Pi_\lambda(f(\Pi_{\text{CRS}}(t_1), \dots, \Pi_{\text{CRS}}(t_n))) = f\ \Pi_\lambda(\Pi_{\text{CRS}}(t_1)) \dots \Pi_\lambda(\Pi_{\text{CRS}}(t_n)) \overset{ih}{=} f\ t_1 \dots t_n$
- If $u = \Lambda\ \lambda x.t$ then $\Pi_\lambda(\Pi_{\text{CRS}}(\Lambda\ \lambda x.t)) = \Pi_\lambda([x]\Pi_{\text{CRS}}(t)) = \Lambda\ \lambda x.\Pi_\lambda(\Pi_{\text{CRS}}(t)) \overset{ih}{=} \Lambda\ \lambda x.t$

For the second equality we proceed by structural induction on the CRS-metaterm A.
Base case: If $A = x$ then $\Pi_{\text{CRS}}(\Pi_\lambda(x)) = \Pi_{\text{CRS}}(x) = x$. The case $A = f$ is similar.
Induction:

- If $A = f(A_1, \dots, A_n)$ we obtain $\Pi_{\text{CRS}}(\Pi_\lambda(f(A_1, \dots, A_n))) = \Pi_{\text{CRS}}(f\ \Pi_\lambda(A_1) \dots \Pi_\lambda(A_n)) = f(\Pi_{\text{CRS}}(\Pi_\lambda(A_1)), \dots, \Pi_{\text{CRS}}(\Pi_\lambda(A_n)) \overset{ih}{=}\ f(A_1, \dots, A_n)$.
- We proceed similarly if $A = Z(A_1, \dots, A_n)$.
- If $A = [x]A'$ we obtain $\Pi_{\text{CRS}}(\Pi_\lambda([x]A')) = \Pi_{\text{CRS}}(\Lambda\ \lambda x.\Pi_\lambda(A')) = [x]\Pi_{\text{CRS}}(\Pi_\lambda(A')) \overset{ih}{=}\ [x]A'$

These properties of the projection functions are used to prove that, given a CRS matching-equation, the solution of the translated matching-equation in the λ-calculus can be projected into a CRS assignment that is a solution of the initial matching-equation in the CRS.

THEOREM 25 (Soundness). *Let $L \in \mathcal{MT}$ be a closed* CRS*-pattern and $t \in \mathcal{T}_{\text{CRS}}$ be a a* CRS*-term. Then:*

$$\sigma \in \mathcal{S}ol(\Pi_\lambda(L) \ll_{\beta\eta} \Pi_\lambda(t)) \quad \Rightarrow \quad \Pi_{\text{CRS}}(\sigma) \in \mathcal{S}ol(L \ll_{\text{CRS}} t)$$

Proof. According to Definitions 14 and 15 we have to prove that

$$\sigma(\Pi_\lambda(L)) =_{\beta\eta} \Pi_\lambda(t) \quad \Rightarrow \quad \Pi_{\text{CRS}}(\sigma)(L) \equiv t$$

and since $\Pi_\lambda(t)$ is in long $\beta\eta$-normal form (Proposition 17) then we should prove that

$$\sigma(\Pi_\lambda(L)) \mapsto\!\!\!\twoheadrightarrow_\beta \Pi_\lambda(t) \quad \Rightarrow \quad \Pi_{\text{CRS}}(\sigma)(L) \equiv t$$

If L contains no CRS-metavariables then $\Pi_\lambda(L)$ contains no free variables. Consequently, σ acts as the identity and so, according to Definition 19 and since Π_{CRS} preserves the domain of the substitution, $\Pi_{\text{CRS}}(\sigma)$ acts as the identity as well. Since $\Pi_\lambda(L) \equiv \Pi_\lambda(t)$ and since Π_λ is a bijection then $L \equiv t$.

We should consider next the case where $Z_1, \dots, Z_n$ are the metavariables of the term $L_{\lceil Z_1(y_1,\dots,y_{k_1})\rceil_{\omega_1} \dots \lceil Z_n(y_1,\dots,y_{k_n})\rceil_{\omega_n}}$ with $\omega_1, \dots, \omega_n$ sharing no prefix. Since $\mathcal{FV}(\Pi_\lambda(L)) = \{Z_1, \dots, Z_n\}$ the substitution σ is of the form $\sigma = \{Z_1/\xi_1, \dots, Z_n/\xi_n\}$ and since there is no interference between the n metavariable positions then by Lemma 23(2), $\sigma(\Pi_\lambda(L_{\lceil Z_1(y_1,\dots,y_{k_1})\rceil_{\omega_1} \dots \lceil Z_n(y_1,\dots,y_{k_n})\rceil_{\omega_n}})) = \sigma(\Pi_\lambda(L)_{\lceil Z_1\ y_1\ \dots\ y_{k_1}\rceil_{\pi_\lambda(\omega_1,L)} \dots \lceil Z_n\ y_1\ \dots\ y_{k_n}\rceil_{\pi_\lambda(\omega_n,L)}})$ and this latter term is equivalent to $\sigma(\Pi_\lambda(L))_{\lceil \xi_1\ y_1\ \dots\ y_{k_1}\rceil_{\pi_\lambda(\omega_1,L)} \dots \lceil \xi_n\ y_1\ \dots\ y_{k_n}\rceil_{\pi_\lambda(\omega_n,L)}}$. Since the translation does not generate applications involving the sub-terms at the n positions, all we need to conclude the proof is the property for only one metavariable.

We have thus to prove that if $\sigma(\Pi_\lambda(L_{\lceil Z(y_1,\ldots,y_k)\rceil_\omega})) \mapsto\!\!\!\!\twoheadrightarrow_\beta \Pi_\lambda(t)$ then $\Pi_{\text{CRS}}(\sigma)(L_{\lceil Z(y_1,\ldots,y_k)\rceil_\omega}) \equiv t$. Since $\sigma = \{Z/\xi\}$ then we can use Lemma 23(2) and obtain $\sigma(\Pi_\lambda(L_{\lceil Z(y_1,\ldots,y_k)\rceil_\omega})) = \sigma(\Pi_\lambda(L)_{\lceil Z\ y_1\ \ldots\ y_k\rceil_{\pi_\lambda(\omega,L)}}) = \Pi_\lambda(L)_{\lceil\sigma(Z\ y_1\ \ldots\ y_k)\rceil_{\pi_\lambda(\omega,L)}} \mapsto\!\!\!\!\twoheadrightarrow_\beta \Pi_\lambda(L)_{\lceil(\xi\ y_1\ \ldots\ y_k)\downarrow_\beta\rceil_{\pi_\lambda(\omega,L)}} = \Pi_\lambda(t)$. Since ξ is in long $\beta\eta$-normal form and has the same type as $Z : \tau \rightarrowtriangle \ldots \rightarrowtriangle \tau$ (n type arrows), then ξ has the form $\lambda x_1.\ldots.x_k.u$ with u in normal form and thus $\Pi_\lambda(t) = \Pi_\lambda(L)_{\lceil(u\{x_1/y_1,\ldots,x_k/y_k\})\downarrow_\beta\rceil_{\pi_\lambda(\omega,L)}} = \Pi_\lambda(L)_{\lceil u\{x_1/y_1,\ldots,x_k/y_k\}\rceil_{\pi_\lambda(\omega,L)}}$ and $\Pi_{\text{CRS}}(\sigma) = \{(Z, \Pi_{\text{CRS}}(\xi))\} = \{(Z, \underline{\lambda}x_1.\ldots.x_n.\Pi_{\text{CRS}}(u))\}$. Moreover, since u is in normal form it is easy to see that $\Pi_{\text{CRS}}(u)$ is in normal form as well and that $(\Pi_{\text{CRS}}(u)\{x_1/y_1,\ldots,x_k/y_k\})\downarrow_{\underline{\beta}} = \Pi_{\text{CRS}}(u)\{x_1/y_1,\ldots,x_k/y_k\} = \Pi_{\text{CRS}}(u\{x_1/y_1,\ldots,x_k/y_k\})$. Then we have

$$
\begin{aligned}
\Pi_{\text{CRS}}(\sigma)(L_{\lceil Z(y_1,\ldots,y_k)\rceil_\omega}) &= L_{\lceil(\Pi_{\text{CRS}}(\sigma)(Z))(y_1,\ldots,y_k)\rceil_\omega}\\
&\mapsto\!\!\!\!\twoheadrightarrow_{\underline{\beta}} L_{\lceil(\Pi_{\text{CRS}}(u)\{x_1/y_1,\ldots,x_k/y_k\})\downarrow_{\underline{\beta}}\rceil_\omega}\\
&= L_{\lceil\Pi_{\text{CRS}}(u\{x_1/y_1,\ldots,x_k/y_k\})\rceil_\omega}\\
&= \Pi_{\text{CRS}}(\Pi_\lambda(L_{\lceil\Pi_{\text{CRS}}(u\{x_1/y_1,\ldots,x_k/y_k\})\rceil_\omega}))\ \text{(by Lemma 24(2))}\\
&= \Pi_{\text{CRS}}(\Pi_\lambda(L)_{\lceil\Pi_\lambda(\Pi_{\text{CRS}}(u\{x_1/y_1,\ldots,x_k/y_k\}))\rceil_{\pi_\lambda(\omega,L)}})\ \text{(by Lemma 23(2))}\\
&= \Pi_{\text{CRS}}(\Pi_\lambda(L)_{\lceil u\{x_1/y_1,\ldots,x_k/y_k\}\rceil_{\pi_\lambda(\omega,L)}})\ \text{(by Lemma 24(1))}\\
&= \Pi_{\text{CRS}}(\Pi_\lambda(t))\\
&= t\ \text{(by Lemma 24(2))}.
\end{aligned}
$$

The other direction of the implication also holds: a solution of a matching-equation in the CRS is also a solution of the matching-equation in the λ-calculus, after the respective translations.

THEOREM 26 (Completeness). *Let $L \prec\!\!\prec_{\text{CRS}} t$ be a* CRS *matching-equation with $L \in \mathcal{MT}$ a closed pattern and $t \in \mathcal{T}_{\text{CRS}}$, and let σ be a* CRS *assignment. Then:*

$$\sigma \in \mathcal{S}ol(L \prec\!\!\prec_{\text{CRS}} t) \quad\Rightarrow\quad \Pi_\lambda(\sigma) \in \mathcal{S}ol(\Pi_\lambda(L) \prec\!\!\prec_{\beta\eta} \Pi_\lambda(t))$$

Proof. The proof method is similar to the one used in Theorem 25. We need to prove that

$$\sigma L \equiv t \quad\Rightarrow\quad \Pi_\lambda(\sigma)(\Pi_\lambda(L)) \mapsto\!\!\!\!\twoheadrightarrow_\beta \Pi_\lambda(t)$$

If L contains no CRS-metavariables, then σ acts as the identity and so (by Definition 16 that preserves the domain of the substitution) $\Pi_\lambda(\sigma)$ acts as the identity as well. Since $L \equiv t$ then $\Pi_\lambda(L) = \Pi_\lambda(t)$.

We suppose next that $Z_1,\ldots,Z_n$ are the n metavariables of the metaterm $L_{\lceil Z_1(y_1,\ldots,y_{k_1})\rceil_{\omega_1}\ldots\lceil Z_n(y_1,\ldots,y_{k_n})\rceil_{\omega_n}}$ with $\omega_1,\ldots,\omega_n$ sharing no prefix. Since there is no interference between the n metavariable positions, using a similar reasoning as in Theorem 25, we need to prove the property for only one metavariable Z of arity k at the position ω in L to conclude the proof. By Definition 9, σ has the form $\{(Z, \underline{\lambda}x_1.\ldots.x_k.u)\}$, where u is a CRS-term, and thus $\Pi_\lambda(\sigma) = \{Z/\lambda x_1.\ldots.x_k.\Pi_\lambda(u)\}$ where the term $\Pi_\lambda(u) : \tau$ is in long $\beta\eta$-normal form. By hypothesis we have $\sigma(L_{\lceil Z(y_1,\ldots,y_k)\rceil_\omega}) \mapsto\!\!\!\!\twoheadrightarrow_{\underline{\beta}} L_{\lceil u\{x_1/y_1,\ldots,x_k/y_k\}\rceil_\omega} = t$ and thus we have to prove that $\Pi_\lambda(\sigma)(\Pi_\lambda(L_{\lceil Z(y_1,\ldots,y_k)\rceil_\omega})) \mapsto\!\!\!\!\twoheadrightarrow_\beta \Pi_\lambda(t)$. We have

$$
\begin{aligned}
\Pi_\lambda(\sigma)(\Pi_\lambda(L_{\lceil Z(y_1,\ldots,y_k)\rceil_\omega})) &= \Pi_\lambda(\sigma)(\Pi_\lambda(L)_{\lceil Z y_1\ldots y_k\rceil_{\pi_\lambda(\omega,L)}}) \quad \text{(by Lemma 23(2))}\\
&= \Pi_\lambda(L)_{\lceil\Pi_\lambda(\sigma)(Z)y_1\ldots y_k\rceil_{\pi_\lambda(\omega,L)}}\\
&\mapsto\!\!\!\!\twoheadrightarrow_\beta \Pi_\lambda(L)_{\lceil\Pi_\lambda(u)\{x_1/y_1,\ldots,x_k/y_k\}\downarrow_\beta\rceil_{\pi_\lambda(\omega,L)}}\\
&= \Pi_\lambda(L)_{\lceil\Pi_\lambda(u\{x_1/y_1,\ldots,x_k/y_k\})\rceil_{\pi_\lambda(\omega,L)}}\\
&= \Pi_\lambda(L_{\lceil u\{x_1/y_1,\ldots,x_k/y_k\}\rceil_\omega})\ \text{(by Lemma 24(2))}\\
&= \Pi_\lambda(t)
\end{aligned}
$$

COROLLARY 27. *The* CRS *matching problem between a* CRS*-pattern and a* CRS*-term is decidable and unitary.*

5. Translating CRS into the Rewriting Calculus

We propose in this section a translation of CRS-(meta)terms into ρ-terms and we show that to every CRS-reduction we can associate a corresponding ρ-reduction. The assignment application used for performing term reductions in CRSs (and thus the matching the CRS-reduction relies on) is based on the λ-calculus. Consequently, to shallowly encode CRSs into the rewriting calculus the target ρ-calculus should use a matching congruence more complex than the syntactic one. For this reason we choose $\rho Cal_{\P}$ described in Section 2.4 as target calculus.

5.1. The translation

In the following we suppose that the set of constants of $\rho Cal_{\P}$ is the same as the set of functional symbols of CRSs and the set of variables of $\rho Cal_{\P}$ is the union of the sets of variables and metavariables of CRSs.

DEFINITION 28 (Translation). *The translation $\mathcal{E}$ of a* CRS*-metaterm t into a term of $\rho Cal_{\P}$, denoted by $\overline{t}$, is inductively defined as follows:*

- CRS*-metaterms into ρ-terms*

$$\begin{array}{lcl@{\qquad\qquad}rcl}
\overline{x} & \triangleq & x & \overline{f(t_1,\ldots,t_n)} & \triangleq & f(\overline{t_1},\ldots,\overline{t_n}) \\
\overline{[x]t} & \triangleq & x \rightarrowtriangle \overline{t} & \overline{Z(t_1,\ldots,t_n)} & \triangleq & Z\,\overline{t_1}\ldots\overline{t_n}
\end{array}$$

- CRS *rewrite rules into ρ-terms:*

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

- CRS *substitutes into ρ-terms:*

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

- CRS *assignments into ρ-substitutions:*

$$\overline{\{\ldots,(Z,\xi),\ldots\}} \triangleq \{\ldots, Z/\overline{\xi},\ldots\}$$

One can notice that the CRS-abstraction operator $[\_]\_$ is translated into a ρ-abstraction. Moreover, since in ρ-calculus rewrite rules are first class objects, a CRS rewrite rule is translated into a ρ-term and more precisely into a ρ-rule. The translation of the abstraction operator and of the rewrite rules of CRSs using the same abstraction operator of ρ-calculus corresponds to the uniform treatment of first and higher-order rewriting in the ρ-calculus. An n-ary CRS-metavariable (function) corresponds in $\rho Cal_{\P}$ to a variable (constant) applied to n ρ-terms.

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

EXAMPLE 29. *We have already seen how the β-rule of the λ-calculus can be translated into a* CRS*-rule (see Example 12). When translating this* BetaCRS *rule into the rewriting calculus the following term is obtained:*

$$\overline{\textsc{BetaCRS}} = \overline{App(Ab([x]Z(x)),Z_1) \to Z(Z_1)} \triangleq App(Ab(x \rightarrowtriangle (Z\;x)),Z_1) \rightarrowtriangle Z\,Z_1$$

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

Since there is no application symbol in the syntax of CRSs, the CRS-rules are never directly applied to a CRS-term. Nevertheless the CRS-rule is translated into a ρ-abstraction ensuring that the corresponding variables are bound in the translation and thus, that the pattern condition is preserved by the translation.

PROPOSITION 30. *If a* CRS*-metaterm L is a* CRS*-pattern, then $\overline{L}$ is a ρ-pattern.*

Moreover, as a consequence of the definition of the translation, the ρ-terms obtained from CRS-metaterms contain no redexes and thus, are in normal form.

As for the λ-calculus and CRS, 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 CRS-terms and ρ-terms. We use for the position transformations in a ρ-term a function π_ρ defined in the same way as the function π_λ in Section 4.2 (Definition 18).

DEFINITION 31. *Let A be a* CRS*-metaterm and $\omega \in \mathcal{P}os(A)$ a position in A. The function π_ρ is inductively defined by:*

- $\pi_\rho(\epsilon, A) = \epsilon$,
- $\pi_\rho((n-i).\omega, Z(A_1, \ldots, A_n)) = 1^i.2.\pi_\rho(\omega, A_{(n-i)})$, *where* $i \in \{0, \ldots, n-1\}$,
- $\pi_\rho((n-i).\omega, f(A_1, \ldots, A_n)) = 1^i.2.\pi_\rho(\omega, A_{(n-i)})$, *where* $i \in \{0, \ldots, n-1\}$,
- $\pi_\rho(1.\omega, [x]A) = 1.\pi_\rho(\epsilon, x)$,
- $\pi_\rho(2.\omega, [x]A) = 2.\pi_\rho(\omega, A)$.

5.2. Examples

In what follows we give some examples of CRS-reductions as well as the corresponding reductions in $\rho Cal_{¶}$. We show in the next section that all CRS-derivations can be expressed in the rewriting calculus using the translation we have proposed above.

EXAMPLE 32. *We consider the* CRS*-reduction $f(App(Ab([x]f(x)), a)) \mapsto\!\!\!\rightarrow_{\text{BetaCRS}} f(f(a))$ presented in Example 13. We should point out that in this reduction the* CRS*-rule is not applied at the head position of the considered term but deeper in the term. In the rewriting calculus the application operator is explicit and thus, to obtain a similar reduction in $\rho Cal_{¶}$, we have to use an explicit congruence rule that pushes the application of the rule we want to apply down in the term. Therefore, we apply the ρ-term $f(y) \twoheadrightarrow f(\overline{\text{BetaCRS}}\ y)$ (with* BetaCRS *defined in Example 12) to the translation of the initial* CRS*-term and we obtain the following reduction:*

$$\begin{array}{ll}
& (f(y) \twoheadrightarrow f(\overline{\text{BetaCRS}}\ y))\ (\overline{f(App(Ab([x]f(x)), a))}) \\
\mapsto_\rho & f(\overline{\text{BetaCRS}}\ y)[f(y) \ll f(App(Ab(x \twoheadrightarrow f(x)), a))] \\
\mapsto_\sigma & f(\overline{\text{BetaCRS}}\ App(Ab(x \twoheadrightarrow f(x)), a)) \\
= & f((App(Ab(x \twoheadrightarrow (Z\ x)), Z_1) \twoheadrightarrow Z\ Z_1)\ App(Ab(x \twoheadrightarrow f(x)), a)) \\
\mapsto_\rho & f(Z\ Z_1[App(Ab(x \twoheadrightarrow (Z\ x)), Z_1) \ll App(Ab(x \twoheadrightarrow f(x)), a)]) \\
\mapsto_\sigma & f(Z\ Z_1\{Z/z \twoheadrightarrow f(z), Z_1/a\}) = f((z \twoheadrightarrow f(z))\ a) \\
\mapsto_\rho & f(f(z)[z \ll a]) \\
\mapsto_\sigma & f(f(a))
\end{array}$$

One can notice in Example 32 that the reductions in CRSs and in the rewriting calculus lead to the same final term, modulo the translation, but we do not have a 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 ρ-calculus). The same behavior is obtained in the following (more elaborated) example.

EXAMPLE 33 (λ-calculus with surjective pairing). *Given a* CRS *with the following set of rewrite rules $\mathcal{R}$:*

$$\begin{aligned}\mathcal{R} \ = \ \{ \ & \wp_0 : P_0(P(X_1, X_2)) \to X_1, \\ & \wp_1 : P_1(P(X_1, X_2)) \to X_2, \\ & \wp : P(P_0 X_1, P_1 X_1) \to X_1, \\ & \textsc{BetaCRS} \ \}\end{aligned}$$

where $X_1, X_2 \in \mathcal{Z}_0$, $P \in \mathcal{F}_2$ (pair function), $P_0, P_1 \in \mathcal{F}_1$ (projections) and BetaCRS *as in Example 12.*

We consider the CRS*-term $t = App(Ab([z]P(P_1 z, P_0 z)), P(x_1, x_2))$ where $Ab([z]P(P_1 z, P_0 z))$ intuitively swaps the elements of a pair. One can check that the assignment $\sigma = \{(Z, \underline{\lambda} z.P(P_1 z, P_0 z), (Z_1, P(x_1, x_2))\}$ can be used to reduce the* CRS*-term t using the rule* BetaCRS*:*

$$\sigma(Z(Z_1)) = (\sigma(Z))(\sigma(Z_1)) = (\underline{\lambda} z.P(P_1 z, P_0 z))(P(x_1, x_2))\downarrow_{\underline{\beta}} = P(P_1(P(x_1, x_2)), P_0(P(x_1, x_2)))$$

Next we can apply the rules $\wp_1$ and $\wp_0$ to the first and second arguments of P respectively, using the assignment $\sigma' = \{(X_1, x_1), (X_2, x_2)\}$ and we obtain the final result:

$$P(P_1(P(x_1, x_2)), P_0(P(x_1, x_2))) \mapsto_{\wp_1} P(x_2, P_0(P(x_1, x_2))) \mapsto_{\wp_0} P(x_2, x_1)$$

When we translate the above set of CRS *rewrite rules into $\rho Cal_{\P}$ we obtain*

$$\begin{aligned}\overline{\wp_0} \ &\triangleq \ P_0(P(X_1, X_2)) \rightarrowtriangle X_1 \\ \overline{\wp_1} \ &\triangleq \ P_1(P(X_1, X_2)) \rightarrowtriangle X_2 \\ \overline{\wp} \ &\triangleq \ P(P_0 X_1, P_1 X_1) \rightarrowtriangle X_1 \\ \overline{\textsc{BetaCRS}} \ &\triangleq \ App(Ab(x \rightarrowtriangle (Z\ x)), Z_1) \rightarrowtriangle Z\ Z_1\end{aligned}$$

and the translation of the initial CRS*-term t is:*

$$\overline{t} \triangleq App(Ab(z \rightarrowtriangle P(P_1 z, P_0 z)), P(x_1\ x_2))$$

Starting from the CRS *reduction above, we build the ρ-terms u_1 and u_2 corresponding to the applications of the rules $\overline{\textsc{BetaCRS}}$ and $\overline{\wp_0}, \overline{\wp_1}$ respectively. As in Example 32 we have to use an explicit congruence rule to simulate the application of the last two rules and we obtain the ρ-terms*

$$u_1 = \overline{\textsc{BetaCRS}} \text{ and } u_2 = P(x, y) \rightarrowtriangle P(\overline{\wp_1}\ x, \overline{\wp_0}\ y)$$

The ρ-terms u_1, u_2 are called derivation trace terms *(see Theorem 39) and are used to apply the translation of the* CRS*-rules at the correct positions in the term. Starting from these terms we build the ρ-term $u_2\ (u_1\ \overline{t})$. When reducing the term $u_1\ \overline{t}$ we use the substitution $\overline{\sigma} = \{Z/z \rightarrowtriangle P(P_1 z, P_0 z), Z_1/P(x_1\ x_2)\}$ and since*

$$\begin{aligned}\overline{\sigma}(Z\ Z_1) \ &= \ \overline{\sigma}(Z)\ \overline{\sigma}(Z_1) = (z \rightarrowtriangle P(P_1 z, P_0 z))\ P(x_1, x_2) \\ &\mapsto_{\rho} \ P(P_1 z, P_0 z)[z \ll P(x_1, x_2)] \\ &\mapsto_{\sigma} \ P(P_1(P(x_1, x_2)), P_0(P(x_1, x_2)))\end{aligned}$$

we obtain

$$\begin{aligned}u_1\,\overline{t} \ &= \ (App(Ab(x \rightarrowtriangle (Z\ x)), Z_1) \rightarrowtriangle Z\ Z_1)\ App(Ab(z \rightarrowtriangle P(P_1 z, P_0 z)), P(x_1, x_2)) \\ &\mapsto_{\rho} \ Z\ Z_1[App(Ab(x \rightarrowtriangle (Z\ x)), Z_1)) \ll App(Ab(z \rightarrowtriangle P(P_1 z, P_0 z)), P(x_1, x_2))] \\ &\mapsto_{\sigma} \ \overline{\sigma}(Z\ Z_1) \\ &\mapsto_{\rho\sigma\delta} \ P(P_1(P(x_1, x_2)), P_0(P(x_1, x_2)))\end{aligned}$$

The application of u_2 to this intermediate result can be then reduced as follows:

$$\begin{array}{rll} u_2\,(u_1\,\bar{t}) & \mapsto_{\rho\sigma\delta} & (P(x,y) \rightarrowtriangle P(\overline{\wp_1}\ x, \overline{\wp_0}\ y))\ P(P_1(P(x_1,x_2)), P_0(P(x_1,x_2))) \\ & \mapsto_{\rho} & P(\overline{\wp_1}\ x, \overline{\wp_0}\ y)[P(x,y) \ll P(P_1(P(x_1,x_2)), P_0(P(x_1,x_2)))] \\ & \mapsto_{\sigma} & P(\overline{\wp_1}\ P_1(P(x_1,x_2)),\ \overline{\wp_0}\ P_0(P(x_1,x_2))) \end{array}$$

The last reduction consists in applying the ρ-rules $\overline{\wp_0}$ and $\overline{\wp_1}$ using the substitution $\overline{\sigma'} = \{X_1/x_1, X_2/x_2\}$ and thus

$$u_2\,(u_1\,\bar{t}) \mapsto_{\rho\sigma\delta} P(\overline{\wp_1}\ P_1(P(x_1,x_2)),\ \overline{\wp_0}\ P_0(P(x_1,x_2))) \mapsto\!\!\!\rightarrow_{\rho\sigma\delta} P(x_2,x_1)$$

which is the translation of the result of the original CRS *reduction.*

EXAMPLE 34 (Summation). *We consider the finite sum $\Sigma_{i=0}^{n} s(i)$ where n is a natural number and s the successor function. It can be expressed as the following* CRS *rewrite rules set:*

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

where $0 \in \mathcal{F}_0$, $s \in \mathcal{F}_1$, $\Sigma, a \in \mathcal{F}_2$, $X,Y,N \in \mathcal{Z}_0$ and $F \in \mathcal{Z}_1$. 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 summation in a recursive way.*

We consider the sum

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

The corresponding CRS*-term is $t = \Sigma(s(0),[i]s(i))$. We show next that reducing t we obtain the expected result $s(s(s(0)))$. We denote by L_i and R_i the left-hand side and the right-hand side of the rule Rec_i, for $i = 0,1$. To perform the reduction of t we first apply the rule Rec_1 with the assignment $\sigma_1 = \{(N,0),(F,\underline{\lambda}y.s(y))\}$. We have*

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

and thus, modulo α-conversion, $\sigma_1(L_1) = t$. When instantiating the right-hand side we obtain

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

Next we apply the CRS*-rule Rec_0 to the sub-term $t_0 = \Sigma(0,[x]s(x))$ since the assignment $\sigma_0 = \{(F,\underline{\lambda}y.s(y))\}$ is a solution of the matching between L_0 and t_0:*

$$\sigma_0(L_0) = \sigma_0\ (\Sigma(0,[x]F(x))) = \Sigma(0,[x]\sigma_0(F(x))) = \Sigma(0,[x](\underline{\lambda}y.s(y))(x))\downarrow_{\underline{\beta}} = \Sigma(0,[x]s(x)) = t_0$$

When applying σ_0 to the right-hand side of the rule we obtain

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

and therefore, the following reduction is obtained for the whole CRS*-term t:*

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

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

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

Now, we translate into $\rho Cal_{\P}$ the set $\mathcal{R}$ of CRS *rewrite rules:*

$$\begin{array}{rcl}
\overline{S_0} & \triangleq & a(0,Y) \mathrel{-\!\triangleright} Y \\
\overline{S_1} & \triangleq & a(s(X),Y) \to s(a(X,Y)) \\
\overline{Rec_0} & \triangleq & \Sigma(0, x \mathrel{-\!\triangleright} (F\ x)) \mathrel{-\!\triangleright} F\ 0 \\
\overline{Rec_1} & \triangleq & \Sigma(s(N), x \mathrel{-\!\triangleright} (F\ x)) \mathrel{-\!\triangleright} a(\Sigma(N, x \mathrel{-\!\triangleright} (F\ x)), F\ s(N))
\end{array}$$

and the CRS*-term t:*

$$\overline{t} \triangleq \Sigma(s(0), i \mathrel{-\!\triangleright} s(i))$$

We proceed as in Example 33 and starting from the reduction shown above we build the ρ-terms

$$u_1 = \overline{Rec_1} \text{ and } u_2 = (a(x,y) \mathrel{-\!\triangleright} a(\overline{Rec_0}\ x,\ y))$$

corresponding to the application of the rules $\overline{Rec_1}$ and $\overline{Rec_0}$ respectively and we built the ρ-term

$$u_2\ (u_1\ \overline{t})$$

We first reduce the application of $\overline{Rec_1}$ to $\overline{t}$ using the substitution $\overline{\sigma_1} = \{N/0,\ F/y \mathrel{-\!\triangleright} s(y)\}$:

$$\begin{array}{rcl}
\overline{\sigma_1}\ (a(\Sigma(N, x \mathrel{-\!\triangleright} (F\ x)), F\ s(N))) & = & a(\Sigma(0, x \mathrel{-\!\triangleright} (y \mathrel{-\!\triangleright} s(y))\ x), (y \mathrel{-\!\triangleright} s(y))\ s(0)) \\
& \mapsto\!\!\!\!\twoheadrightarrow_{\rho\sigma\delta} & a(\Sigma(0, x \mathrel{-\!\triangleright} s(x)), s(s(0)))
\end{array}$$

The term u_2 is then applied to this intermediate result:

$$\begin{array}{rcl}
u_2\, a(\Sigma(0, x \mathrel{-\!\triangleright} s(x)), s(s(0))) & = & (a(x,y) \mathrel{-\!\triangleright} a(\overline{Rec_0}\ x, y))\ a(\Sigma(0, x \mathrel{-\!\triangleright} s(x)), s(s(0))) \\
& \mapsto_{\rho} & a(\overline{Rec_0}\ x, y)[a(x,y) \ll a(\Sigma(0, x \mathrel{-\!\triangleright} s(x)), s(s(0)))] \\
& \mapsto_{\sigma} & a(\overline{Rec_0}\ \Sigma(0, x \mathrel{-\!\triangleright} s(x)), s(s(0)))
\end{array}$$

The substitution $\overline{\sigma_2} = \{F/y \mathrel{-\!\triangleright} s(y)\}$ is used for the application of the rule $\overline{Rec_0}$ and since

$$\overline{\sigma_2}(F\ 0) = (y \mathrel{-\!\triangleright} s(y))\ 0\ \ \mapsto_{\rho\sigma}\ \ s(0)$$

we obtain

$$u_2\ (u_1\ \overline{t}) \mapsto\!\!\!\!\twoheadrightarrow_{\rho\sigma\delta} a(\Sigma(0, x \mathrel{-\!\triangleright} s(x)), s(s(0))) \mapsto\!\!\!\!\twoheadrightarrow_{\rho\sigma\delta} a(s(0), s(s(0)))$$

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

$$\overline{S_0}\ (\overline{S_1} a(s(0), s(s(0)))) \mapsto\!\!\!\!\twoheadrightarrow_{\rho\sigma\delta} \overline{S_0}\ s(a(0, s(s(0))) \mapsto\!\!\!\!\twoheadrightarrow_{\rho\sigma\delta} s(s(s(0)))$$

EXAMPLE 35 (Recursion). *Given a* CRS *with the following set of rewrite rules $\mathcal{R}$:*

$$\begin{array}{rcl}
\mathcal{R} & = & \{\ \ R_0 : rec(0, A, [x][y]F(x,y)) \to A\ ; \\
& & \quad\ R_1 : rec(s(N), A, [x][y]F(x,y)) \to F(N, rec(N, A, [x][y]F(x,y))\ \}
\end{array}$$

where $rec \in \mathcal{F}_3$, $s \in \mathcal{F}_1$, $A, N \in \mathcal{Z}_0$ and $F \in \mathcal{Z}_2$.

We consider the CRS*-term $t = rec(s(0), s(s(0), [x][y]s(y))$ representing the addition of the two natural numbers $s(0)$ and $s(s(0))$ and we reduce the term t to the final result $s(s(s(0)))$. For this, we first apply to the* CRS*-term t the* CRS *rewrite rule R_1 using the assignment $\sigma_1 = \{(N,0), (A, s(s(0)), (F, \underline{\lambda} z_1 z_2.s(z_2))\}$ and we obtain*

$$\begin{array}{rcl}
t & \mapsto_{R_1} & \sigma_1\ (F(N, rec(N, A, [x][y]F(x,y))) \\
& = & (\sigma_1(F))(\sigma_1(N), rec(\sigma_1(N), \sigma_1(A), [x][y]\sigma_1(F(x,y))) \\
& = & (\underline{\lambda} z_1 z_2.s(z_2))(0, rec(0, s(s(0)), [x][y](\underline{\lambda} z_1 z_2.s(z_2))(x,y))\downarrow_{\underline{\beta}} \\
& = & (\underline{\lambda} z_1 z_2.s(z_2))(0, rec(0, s(s(0)), [x][y]s(y))\downarrow_{\underline{\beta}} \\
& = & s(rec(0, s(s(0), [x][y]s(y))))
\end{array}$$

Next we apply the rule R_0 to the sub-term $rec(0, s(s(0)), [x][y]s(y)))$ with the assignment $\sigma_0 = \{(A, s(s(0)), (F, \underline{\lambda} z_1 z_2.s(z_2))\}$ and we obtain

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

We now translate the example into $\rho Cal_\P$ where the following two ρ-terms correspond to the two CRS *rewrite rules*

$$\begin{array}{rcl} \overline{R_0} & \triangleq & rec(0, A, x \rightarrowtriangle (y \rightarrowtriangle (F(x, y)))) \ \rightarrowtriangle \ A \\ \overline{R_1} & \triangleq & rec(s(N), A, x \rightarrowtriangle (y \rightarrowtriangle (F(x, y)))) \rightarrowtriangle F(N, rec(N, A, x \rightarrowtriangle (y \rightarrowtriangle (F(x, y))))) \end{array}$$

The CRS*-term t is translated into the following ρ-term:*

$$\bar{t} \triangleq rec(s(0), s(s(0), x \rightarrowtriangle (y \rightarrowtriangle s(y))))$$

As in the previous examples, starting from the CRS *reduction presented above, we build the following ρ-terms corresponding to the application of the rules $\overline{R_1}$ and $\overline{R_0}$ rules.*

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

and we build the ρ-term: $u_2\ (u_1\ \bar{t})$. We start reducing this ρ-term by applying the ρ-rule $\overline{R_1}$ to $\bar{t}$ using the substitution $\overline{\sigma_1} = \{F/z_1 \rightarrowtriangle z_2 \rightarrowtriangle s(z_2),\ N/0,\ A/s(s(0))\}$ and we obtain:

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

When reducing the application of u_2 to this intermediate result we obtain

$$\begin{array}{rl} & (s(x) \rightarrowtriangle s(\overline{R_0}\ x))\ s(rec(0, s(s(0)), x \rightarrowtriangle (y \rightarrowtriangle s(y)))) \\ \mapsto_\rho & s(\overline{R_0}\ x)[s(x) \ll s(rec(0, s(s(0)), x \rightarrowtriangle (y \rightarrowtriangle s(y))))] \\ \mapsto_\sigma & s(\overline{R_0}\ (rec(0, s(s(0)), x \rightarrowtriangle (y \rightarrowtriangle s(y))))) \end{array}$$

Finally we apply $\overline{R_0}$ using the substitution $\overline{\sigma_0} = \{F/z_1 \rightarrowtriangle z_2 \rightarrowtriangle s(z_2),\ A/s(s(0))\}$ and we obtain the expected result.

$$s(\overline{R_0}\ (rec(0, s(s(0)), x \rightarrowtriangle (y \rightarrowtriangle s(y))))) \mapsto\!\!\!\!\twoheadrightarrow_{\rho\sigma\delta} s(s(s(0)))$$

5.3. Properties

We show in what follows that we can always build a ρ-term encoding a given CRS-derivation. Moreover, we will prove that all ρ-reductions of this ρ-term are correct, *i.e.* they lead to the same final term and this term is the translation of the term obtained when performing the CRS-derivation. We first state some auxiliary lemmas concerning the context stability of the translation function and the correspondence between the substitutions in CRS and ρ-calculus.

First of all, we prove the correctness of the position transformation function *w.r.t.* the translation.

LEMMA 36 (Context stability). *Let $A_{\lceil B \rceil_\omega}$ be a* CRS*-metaterm, then:*

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

Proof. The case where $\omega = \epsilon$ is obvious. For the other cases we proceed by structural induction on the CRS-metaterm $A_{\lceil B \rceil_\omega}$.

– If $A = x$ or $A = f$ then $\omega = \epsilon$ and the lemma holds trivially.

– If $A = [x]s$ then ω is either of the form $\omega = 1.\epsilon$ or of the form $\omega = 2.\omega'$ with ω' a position in s.

1. $\overline{([x]s)_{\lceil x \rceil_{1.\epsilon}}} = (x \to \overline{s})_{\lceil x \rceil_{1.\epsilon}} = \overline{[x]s}_{\lceil \overline{x} \rceil_{1.\epsilon}} = \overline{[x]s}_{\lceil \overline{x} \rceil_{\pi_\rho(1.\epsilon,[x]s)}}$
2. $\overline{([x]s)_{\lceil B \rceil_{2.\omega'}}} = \overline{[x](s_{\lceil B \rceil_{\omega'}})} = x \mathrel{-\!\!\triangleright} \overline{s_{\lceil B \rceil_{\omega'}}}$
$\overset{ih}{=} x \mathrel{-\!\!\triangleright} (\overline{s}_{\lceil \overline{B} \rceil_{\pi_\rho(\omega',s)}}) = (x \mathrel{-\!\!\triangleright} \overline{s})_{\lceil \overline{B} \rceil_{2.\pi_\rho(\omega',[x]s)}} = \overline{[x]s}_{\lceil \overline{B} \rceil_{\pi_\rho(2.\omega',[x]s)}}$

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

$$\begin{aligned}
\overline{Z(A_1, \ldots, A_n)_{\lceil B \rceil_{(n-i).\omega'}}} &= \overline{Z(A_1, \ldots, A_{(n-i)}{}_{\lceil B \rceil_{\omega'}}, \ldots, A_n)} = Z\,\overline{A_1} \ldots \overline{A_{(n-i)}{}_{\lceil B \rceil_{\omega'}}} \ldots \overline{A_n} \\
&\overset{ih}{=} Z\,\overline{A_1} \ldots \overline{A_{(n-i)}}{}_{\lceil \overline{B} \rceil_{\pi_\rho(\omega',A_{(n-i)})}} \ldots \overline{A_n} = (Z\,\overline{A_1} \ldots \overline{A_n})_{\lceil \overline{B} \rceil_{1^i.2.\pi_\rho(\omega',A_{(n-i)})}} \\
&= \overline{Z(A_1, \ldots, A_n)}_{\lceil \overline{B} \rceil_{\pi_\rho((n-i).\omega',Z(A_1,\ldots,A_n))}}
\end{aligned}$$

– If $A = f(A_1, \ldots, A_n)$ then we proceed as in the previous case.

The following lemma states the correctness of the translation with respect to the assignment and substitution applications in CRS and in the rewriting calculus respectively.

LEMMA 37. *Let A be a* CRS*-metaterm, and $t_1, \ldots, t_n$ be* CRS*-terms, and $x_1, \ldots, x_n$ be distinct variables. Then*

$$\overline{A\{x_1/t_1, \ldots, x_n/t_n\}} \;=\; \overline{A}\{x_1/\overline{t_1}, \ldots, x_n/\overline{t_n}\}$$

Proof. We denote $\sigma = \{x_1/t_1, \ldots, x_n/t_n\}$ and $\overline{\sigma} = \{x_1/\overline{t_1}, \ldots, x_n/\overline{t_n}\}$ and we proceed by structural induction on the CRS-metaterm A.

– If $A = x$, with $x \notin \mathit{Dom}(\sigma)$ or $A = f$ then $A\sigma = A$ and the lemma holds trivially. If $A = x_i$ then $\overline{x_i\ \sigma} = \overline{t_i} = \overline{x_i}\ \overline{\sigma}$.

– If $A = [x]s$ then we can suppose, by α-conversion, that $x \notin \mathit{Dom}(\sigma)$ and we have $\overline{([x]s)\sigma}$ $= \overline{[x]s\sigma} = x \to \overline{s\sigma} \overset{ih}{=} x \to \overline{s}\ \overline{\sigma} = (x \to \overline{s})\ \overline{\sigma} = \overline{[x]s}\ \overline{\sigma}$.

– If $A = Z(A_1, \ldots, A_n)$, then $\overline{Z(A_1, \ldots, A_n)\sigma} = \overline{Z\sigma(A_1\sigma, \ldots, A_n\sigma)} = \overline{Z(A_1\sigma, \ldots, A_n\sigma)} =$ $\overline{Z}\ \overline{A_1\sigma} \ldots \overline{A_n\sigma} \overset{ih}{=} \overline{Z}\ \overline{A_1}\overline{\sigma} \ldots \overline{A_n}\overline{\sigma} = (\overline{Z}\ \overline{A_1} \ldots \overline{A_n})\overline{\sigma} = \overline{Z(A_1, \ldots, A_n)}\overline{\sigma}$.

– If $A = f(A_1, \ldots, A_n)$ then we proceed as in the previous case.

We show now that the translation *preserves* the matching solution, *i.e.* for every assignment σ that is a solution of a CRS matching problem, its translation into a ρ-substitution is a solution of the corresponding matching problem in ρ-calculus. As a consequence, given an assignment used for the application of a CRS rewrite rule to a CRS-term, the translation of the rule into $\rho Cal_{¶}$ can be applied to the translation of the corresponding term using as substitution the translation of the CRS assignment.

LEMMA 38. *Let A be a* CRS*-metaterm and σ an assignment. Then*

$$\overline{\sigma}(\overline{A}) \;\mathrel{\mapstochar\twoheadrightarrow}_{\rho_\lambda}\; \overline{\sigma(A)}$$

Proof. By structural induction on the term A.

– If A is a variable x then, since $x \notin \mathcal{MV}$ and thus $x \notin \mathit{Dom}(\overline{\sigma})$, we have $\overline{\sigma}(\overline{x}) = \overline{\sigma}(x) = x$ $= \overline{x} = \overline{\sigma(x)}$.

- If $A = [x]s$ then we can use the induction hypothesis on s and we obtain $\overline{\sigma}(\overline{[x]s}) = \overline{\sigma}(x \rightarrowtriangle \overline{s})$ $= x \rightarrowtriangle (\overline{\sigma}(\overline{s})) \mapsto\!\!\!\!\twoheadrightarrow_{\rho_\lambda} x \rightarrowtriangle \overline{\sigma(s)} = \overline{[x]\sigma(s)} = \overline{\sigma([x]s)}$.

- If $A = f(A_1, \ldots, A_n)$ then we can use the induction hypothesis on the subterms $A_1, \ldots, A_n$:

$$\begin{array}{rcl}
\overline{\sigma}(\overline{f(A_1,\ldots,A_n)}) & = & \overline{\sigma}(f(\overline{A_1},\ldots,\overline{A_n})) = f(\overline{\sigma}(\overline{A_1}),\ldots,\overline{\sigma}(\overline{A_n})) \\
& \mapsto\!\!\!\!\twoheadrightarrow_{\rho_\lambda} & f(\overline{\sigma(A_1)},\ldots,\overline{\sigma(A_n)}) \\
& = & \overline{f(\sigma(A_1),\ldots,\sigma(A_n))} = \overline{\sigma(f(A_1,\ldots,A_n))}
\end{array}$$

- If $A = Z(A_1, \ldots, A_n)$ then we have two cases:

  1. If Z is of arity zero, then $\sigma = \{\ldots,(Z,\xi),\ldots\}$ and ξ contains no $\underline{\beta}$-abstraction. Thus, we have $\overline{\sigma}(\overline{Z}) = \overline{\sigma}(Z) = Z\{Z/\overline{\xi}\} = \overline{\xi} = \overline{\sigma(Z)}$.
  2. If Z is of arity n, then $\sigma = \{\ldots,(Z,\underline{\lambda}x_1\ldots x_n.u),\ldots\}$ and its translation $\overline{\sigma} = \{\ldots,(Z,x_1 \rightarrowtriangle (x_2 \rightarrowtriangle (\ldots(x_n \rightarrowtriangle \overline{u})\ldots))),\ldots\}$. We suppose that $A_1,\ldots,A_n$ contain no occurrences of $x_1\ldots x_n$ which is always possible by α-conversion. We have:

$$\begin{array}{rcl}
\overline{\sigma}(\overline{Z(A_1,\ldots,A_n)}) & = & \overline{\sigma}(Z\,\overline{A_1}\ldots\overline{A_n}) = \overline{\sigma}(Z)\,\overline{\sigma}(\overline{A_1})\ldots\overline{\sigma}(\overline{A_n}) \\
& = & (x_1 \rightarrowtriangle (x_2 \rightarrowtriangle (\ldots(x_n \rightarrowtriangle \overline{u})\ldots)))\,\overline{\sigma}(\overline{A_1})\ldots\overline{\sigma}(\overline{A_n}) \\
& \mapsto\!\!\!\!\twoheadrightarrow_{\rho_\lambda} & \overline{u}\{x_1/\overline{\sigma}(\overline{A_1})\}\ldots\{x_n/\overline{\sigma}(\overline{A_n})\} \\
& = & \overline{u}\{x_1/\overline{\sigma}(\overline{A_1}),\ldots,x_n/\overline{\sigma}(\overline{A_n})\} \quad (\textit{since } \forall i, x_i \notin A_1,\ldots,A_n) \\
& \mapsto\!\!\!\!\twoheadrightarrow_{\rho_\lambda} & \overline{u}\{x_1/\overline{\sigma(A_1)},\ldots,x_n/\overline{\sigma(A_n)}\} \;(\textit{by induction hypothesis}) \\
& = & \overline{u\{x_1/\sigma(A_1),\ldots,x_n/\sigma(A_n)\}} \quad (\textit{by Lemma } 37) \\
& = & \overline{(\underline{\lambda}x_1\ldots x_n.u)(\sigma(A_1),\ldots,\sigma(A_n))\downarrow_{\underline{\beta}}} \\
& = & \overline{(\sigma(Z))(\sigma(A_1),\ldots,\sigma(A_n))} \\
& = & \overline{\sigma(Z(A_1,\ldots,A_n))}
\end{array}$$

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

The congruence $=_{\rho_\lambda}$ is effectively used when the CRS-metaterm A contains metavariables of arity different from zero. In this 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. Therefore, we do not have a syntactical identity between the two terms in the statement of the above lemma, but a reduction from one term to the other.

Using the previous lemmas we can show that starting from a CRS-reduction a corresponding reduction in $\rho Cal_{\P}$ can be obtained and more precisely, a ρ-term encoding the CRS-derivation trace can be constructed. When 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 39. *Given a* CRS*-rule $L \to R$, an assignment σ and two* CRS*-terms $t_{0\lceil\sigma(L)\rceil_\omega}$ and $t_1 \equiv t_0\lceil\omega \hookleftarrow \sigma(R)\rceil$. Given the derivation trace term $u_0 \triangleq \overline{t_0}_{\lceil x\rceil_{\pi_\rho(\omega,t_0)}} \rightarrowtriangle \overline{t_0}_{\lceil \overline{L\to R}x\rceil_{\pi_\rho(\omega,t_0)}}$ then every derivation resulting from $u_0\,\overline{t_0}$ terminates and converges to $\overline{t_1}$:*

$$(u_0\,\overline{t_0}) \mapsto\!\!\!\!\twoheadrightarrow_{\rho\sigma\delta} \overline{t_1}$$

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

$$\begin{aligned}(u_0\,\overline{t_0}) &\triangleq (x \mathrel{-\!\!\triangleright} \overline{L \to R\,x})\,\overline{t_0} \triangleq (x \mathrel{-\!\!\triangleright} \overline{L \to R\,x})\,\overline{\sigma(L)} \triangleq (x \mathrel{-\!\!\triangleright} (\overline{L} \mathrel{-\!\!\triangleright} \overline{R})\,x)\,\overline{\sigma(L)}\\ &\mapsto_{\rho} ((\overline{L} \mathrel{-\!\!\triangleright} \overline{R})\,x)[x \ll \overline{\sigma(L)}] \mapsto_{\sigma} (\overline{L} \mathrel{-\!\!\triangleright} \overline{R})\,\overline{\sigma(L)} \mapsto_{\rho} \overline{R}[\overline{L} \ll \overline{\sigma(L)}] \mapsto_{\sigma} \overline{\sigma}(\overline{R}) \mapsto\!\!\!\!\twoheadrightarrow_{\rho\sigma\delta} \overline{\sigma(R)}\\ &\triangleq \overline{t_1}\end{aligned}$$

In the above reduction we have used the fact that, by Lemma 38, the two matching problems $\overline{L} \mathrel{-\!\!\ll}_{\P} \overline{\sigma(L)}$ and $\overline{L} \mathrel{-\!\!\ll}_{\P} \overline{\sigma}(\overline{L})$ have the same solution $\overline{\sigma}$. Lemma 38 can be extended easily for $\rho Cal_{\P}$ reductions and this version is used for concluding the above reduction.

Another possible reduction is the following one:

$$\begin{aligned}(u_0\,\overline{t_0}) &\triangleq (x \mathrel{-\!\!\triangleright} \overline{L \to R\,x})\,\overline{t_0} \triangleq (x \mathrel{-\!\!\triangleright} \overline{L \to R\,x})\,\overline{\sigma(L)} \triangleq (x \mathrel{-\!\!\triangleright} (\overline{L} \mathrel{-\!\!\triangleright} \overline{R})\,x)\,\overline{\sigma(L)}\\ &\mapsto_{\rho} (x \mathrel{-\!\!\triangleright} \overline{R}[\overline{L} \ll x])\,\overline{\sigma(L)} \mapsto_{\rho} \overline{R}[\overline{L} \ll x][x \ll \overline{\sigma(L)}] \mapsto_{\sigma} \overline{R}[\overline{L} \ll \overline{\sigma(L)}]\\ &\mapsto_{\sigma} \overline{\sigma}(\overline{R}) \mapsto\!\!\!\!\twoheadrightarrow_{\rho\sigma\delta} \overline{\sigma(R)} \triangleq \overline{t_1}\end{aligned}$$

The same arguments as in the previous case have been used in this reduction. These are the only possible derivations since the translations of CRS-terms contain no redexes and the matching problem $\overline{L} \ll_{\P} x$ in the latter derivation has no solution since L cannot be a variable.

The reduction for $\omega \neq \epsilon$ is similar, the only difference being at the matching level. In this case we have to use Lemma 36 to obtain the solution of the matching against the translation of the left-hand side of the rule.

We can notice that we have a longer derivation scheme in $\rho Cal_{\P}$ than in CRSs. For every rewrite step in a CRS, the reduction of the corresponding $\rho Cal_{\P}$-term uses two (ρ)-rule steps plus two (σ)-rule steps for the application of the rewrite rule and some additional $(\rho\sigma\delta)$ steps corresponding to the $\underline{\beta}$-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 $\rho Cal_{\P}$ may involve some derivations but this time performed in ρCal_{λ}.

We can generalize the previous theorem and build, using the derivations of a term t_0 *w.r.t.* a CRS, a ρ-term with a reduction similar to this CRS-derivation.

PROPOSITION 40. *Let t_0 and t_n be two CRS-terms such that $t_0 \mapsto\!\!\!\!\twoheadrightarrow_{\mathcal{R}} t_n$ and $u_0, \ldots, u_n$ the corresponding derivation trace terms in the ρ-calculus. Then every derivation resulting from $(u_n \ldots (u_0\,\overline{t_0}))$ terminates and converges to $\overline{t_n}$.*

Proof. Follows immediately by Theorem 39.

The main difference between the rewriting calculus and CRSs lays in the fact that rewrite rules and consequently their control (application position) are defined at the object-level of ρ-calculus while for the CRSs 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 strongly terminating. Moreover, while for CRSs the $\underline{\beta}$-reduction is implicitly included in the application of the assignment, in $\rho Cal_{\P}$ the corresponding reductions are performed explicitly.

The ρ-terms u_i of Proposition 40 can be built automatically starting from these CRS-reduction steps as stated in Theorem 39. It is obviously interesting to give a method for constructing these terms without knowing *a priori* the derivation steps from t_0 to t_n but only the set of CRS rewrite rules. The development of such a method needs the definition of iteration strategies and of strategies for the generic traversal of terms. This has been done for the initial version of

ρ-calculus either by enriching the calculus with a new operator (Cirstea and Kirchner, 2001) or by adding an "exception handling mechanism" to the calculus (Faure and Kirchner, 2002). 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 ρ-calculus which allows for the definition of iterators and, as a consequence, allows one to represent reductions of first-order rewriting. The use of this method for representing CRS reductions will be the object of a later study.

5.4. Application to Higher-order Rewrite Systems

The Higher-order Rewrite Systems (HRSs), proposed by T. Nipkow in the early nineties (Nipkow, 1991; Nipkow, 1993), were introduced to give a logical framework to programming languages like λ-Prolog and theorem provers like Isabelle. The HRSs consist in an extension of the λ-calculus with algebraic functions, combining the higher-order capabilities of λ-calculus with the expressiveness of first-order rewriting. A HRS $\mathcal{H} = (\mathcal{T}, \mathcal{R})$ is defined by a set of terms $\mathcal{T}$ and a set of rewrite rules $\mathcal{R}$. In (van Oostrom and van Raamsdonk, 1993) the HRS are shown to be as expressive as the CRS, *i.e.* the rewrite relations induced by the two systems are equivalent and similar properties (*e.g.* confluence) are obtained for the two systems. Using the encoding proposed in (van Oostrom and van Raamsdonk, 1993) we can associate a CRS to any HRS and, using the translation defined in this paper, we can encode the corresponding CRS into ρ-calculus. We obtain thus a two-step translation of the HRS into ρ-calculus:

$$\text{HRS} \xrightarrow{\tau} \text{CRS} \xrightarrow{\mathcal{E}} \rho Cal_{\P}$$

where τ is the translation function from HRS to CRS and $\mathcal{E}$ is the function of Definition 28.

Concerning the preservation of the rewrite steps, we recall that HRSs have more "rewriting power" than CRSs, *i.e.* they can do more in one step. This is due to the different definition of the substitution mechanism in the two systems: for the CRSs, the application of an assignment to a term involves only a development on λ-terms, while for the HRSs a reduction to normal form in simply typed λ-calculus is performed. The solution adopted in (van Oostrom and van Raamsdonk, 1993) for preserving rewrite sequences is to add explicitly the BetaCRS rule to the set of CRS-rewrite rules obtained by translating the HRS-rules. This has no particular effect on the ρ-calculus side. The BetaCRS rule is treated as any other CRS-rule and it is translated by the function $\mathcal{E}$ in a term of $\rho Cal_{\P}$ as shown in Example 29.

PROPOSITION 41. *Let t_0, t_n be two HRS-terms such that t_0 rewrites to t_n using the set $\mathcal{R}$ of HRS-rules. If $t'_0 = \mathcal{E}(\tau(t_0))$ and $t'_n = \mathcal{E}(\tau(t_n))$ are the translations of the two HRS-terms into ρ-terms then there exist m ρ-terms $u_0, \ldots, u_m$ such that every derivation resulting from $(u_m\ \ldots\ (u_0\ t'_0))$ terminates and converges to t'_n.*

Proof. By (van Oostrom and van Raamsdonk, 1993) we have that every HRS-rewrite step $t_0 \mapsto_R t_1$, with $R \in \mathcal{R}$, corresponds to a CRS-rewrite sequence $\tau(t_0) \mapsto_{\tau(R)} \mapsto\!\!\!\!\!\to_B{}^! \tau(t_1)$, where $\mapsto\!\!\!\!\!\to_B{}^!$ denotes a reduction to BetaCRS normal form. The ρ-terms $u_0, \ldots, u_m$ can be then built starting from this derivation using the approach described in Theorem 39.

6. Conclusion

The rewriting calculus has an interesting potential in expressing easily the usual computational formalisms. In particular it was shown that it can be used to describe rewrite based languages and object oriented calculi in a natural and simple way. We have shown in this paper that it also allows one to encode higher-order rewriting and in particular we have analyzed the relation with the *Combinatory Reduction systems.* In this context, we have shown that any reduction of a term *w.r.t.* a given CRS can be represented by a corresponding ρ-term.

More precisely, we show that given a CRS-term, a set of CRS-rules and the reduction of this term *w.r.t.* this set of rules, we can build a ρ-term having a corresponding reduction in the rewriting calculus. It would certainly be interesting to build a similar ρ-term using just the initial CRS-term and the CRS-rules. We have previously shown (Cirstea et al., 2004b) how to build ρ-terms describing the application of first-order term rewriting systems *w.r.t.* a given reduction strategy like, for example, innermost or outermost. Intuitively, the ρ-term encoding a first-order rewrite system is a ρ-structure consisting of the corresponding rewrite rules wrapped in an iterator that allows for the repetitive application of the rules. We conjecture that this approach based on (typed object oriented flavored) fixpoints can be applied here for the construction of an appropriate ρ-term only from the set of CRS-rules. Then, the application of the obtained ρ-term to a given term encodes the reduction of the latter *w.r.t.* the set of CRS-rules guided by a given reduction strategy. The ability to typecheck such ρ-terms and thus the encoded rewriting systems and strategies ensures a good trade-off between expressiveness and safety of programs.

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

Since we are mainly interested in the expressive power of the ρ-calculus we have proposed in this paper a translation from CRS to ρ-calculus, but the translation the other way round has not been explicitly defined here. Nevertheless, we believe such a translation is possible but rather complex since the explicit control of rewrite rules in ρ-calculus should be somehow simulated in the corresponding CRS rewrite system.

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

Other higher-order rewrite systems have already been compared. In particular, it has been shown that CRSs and HRSs have the same expressive power and therefore they can be considered equivalent. Using this comparison, we can have an indirect representation of the HRSs in $\rho Cal_{\P}$ that is based on the translation from the HRSs to CRSs and the translation from CRSs to ρ-calculus we have defined in this paper. Some other higher-order systems like the Expression Reduction Systems of Khasidashvili (Khasidashvili, 1990) (ERSs) should be considered. Although CRSs and ERSs are conceptually very similar, their syntax differs in many aspects (for example the restriction in the ERSs to *admissible* assignments). Therefore, it may be interesting to analyze also the correspondence between ρ-calculus and ERSs or other systems like the Explicit Reduction Systems of Pagano (Pagano, 1997).

Acknowledgements The feedback and suggestions of the anonymous referees have been very useful to improve the quality of this paper.

References

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

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

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

Blanqui, F.: 2001a, 'Definitions by rewriting in the Calculus of Constructions'. In: *Logic in Computer Science*. pp. 9–18.

Blanqui, F.: 2001b, 'Type Theory and Rewriting'. Ph.D. thesis, University Paris-Sud.

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

Cirstea, H., G. Faure, and C. Kirchner: 2004a, 'A rho-calculus of explicit constraint application'. In: *Proceedings of WRLA'04*, Vol. 117 of *ENTCS*.

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

Cirstea, H., C. Kirchner, and L. Liquori: 2001a, 'Matching Power'. In: *Proceedings of RTA*, Vol. 2051 of *LNCS*. pp. 77–92, Springer-Verlag.

Cirstea, H., C. Kirchner, and L. Liquori: 2001b, 'The Rho Cube'. In: *Proc. of FOSSACS*, Vol. 2030 of *LNCS*. pp. 166–180.

Cirstea, H., C. Kirchner, and L. Liquori: 2002, 'Rewriting Calculus with(out) Types'. In: F. Gadducci and U. Montanari (eds.): *Proceedings of WRLA'02*. Pisa (Italy), ENTCS.

Cirstea, H., L. Liquori, and B. Wack: 2004b, 'Typed recursion by pattern matching'. In: *Proceeding of the TYPES conference*.

Faure, G. and C. Kirchner: 2002, 'Exceptions in the Rewriting Calculus'. In: *Proc. of RTA*, Vol. 2378 of *LNCS*. pp. 66–82, Springer-Verlag.

Goldfarb, D.: 1981, 'The Undecidability of the Second Order Unification Problem'. *Theoretical Computer Science* **13**, 225–230.

Huet, G.: 1975, 'A Unification Algorithm for Typed Lambda Calculus'. *Theoretical Computer Science* **1**(1), 27–57.

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

Khasidashvili, Z.: 1990, 'Expression Reduction Systems'. In: *Proceedings of I. Vekua Institute of Applied Mathematics*, Vol. 36. pp. 200–220.

Klop, J. W.: 1980, 'Combinatory Reduction Systems'. Ph.D. thesis, CWI.

Klop, J. W., V. v. Oostrom, and F. v. Raamsdonk: 1993, 'Combinatory Reduction Systems: Introduction and Survey'. *Theoretical Computer Science* **121**, 279–308.

Miller, D.: 1991, 'A Logic Programming Language with Lambda-Abstraction, Function Variables, and Simple Unification'. *Journal of Logic and Computation* **1**(4), 497–536.

Miller, D. A. and G. Nadathur: 1986, 'Higher-order logic programming'. In: E. Shapiro (ed.): *Proceedings of the Third International Logic Programming Conference*, Vol. 225 of *LNCS*. pp. 448–462, Springer-Verlag.

Nipkow, T.: 1991, 'Higher-order critical pairs'. In: *Proceedings of Logic in Computer Science*. pp. 342–349.

Nipkow, T.: 1993, 'Orthogonal higher-order rewrite systems are confluent'. In: *Proceedings of the International Conference on Typed La mbda Calculi and Applications*. pp. 306–317.

Nipkow, T. and C. Prehofer: 1998, 'Higher-Order Rewriting and Equational Reasoning'. In: W. Bibel and P. Schmitt (eds.): *Automated Deduction — A Basis for Applications. Volume I: Foundations*. Kluwer.

Pagano, B.: 1997, 'Des calculs de substitution explicites et de leur application la compilation des langages fonctionnels'. Ph.D. thesis, U. Paris VI.

Qian, Z.: 1993, 'Linear Unification of Higher-Order Patterns'. In: M.-C. Gaudel and J.-P. Jouannaud (eds.): *Proceedings of TAPSOFT'93*, Vol. 668 of *Lecture Notes in Computer Science*. pp. 391–405, Springer-Verlag.

van Oostrom, V. and F. van Raamsdonk: 1993, 'Comparing Combinatory Reduction Systems and Higher-order Rewrite Systems'. In: *Higher-Order Algebra, Logic and Term Rewriting (HOA)*. pp. 276–304.

Wolfram, D. A.: 1993, *The Clausal Theory of Types*, Vol. 21 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press.

Contents