# Context-sensitive Conditional Expression Reduction Systems

Zurab Khasidashvili [1]

*School of Information Systems*
*University of East Anglia*
*Norwich NR4 7TJ, England*
*zurab@sys.uea.ac.uk*

Vincent van Oostrom

*Department of Mathematics and Computer Science*
*Vrije Universiteit*
*De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands*
*oostrom@cs.vu.nl*

**Abstract**

We introduce *Context-sensitive Conditional Expression Reduction Systems (CERS)* by extending and generalizing the notion of conditional TRS to the higher order case.

We justify our framework in two ways. First, we define *orthogonality* for CERSs and show that the usual results for orthogonal systems (finiteness of developments, confluence, permutation equivalence) carry over immediately. This can be used e.g. to infer confluence from the subject reduction property in several typed $\lambda$-calculi possibly enriched with pattern-matching definitions.

Second, we express several proof and transition systems as CERSs. In particular, we give encodings of Hilbert-style proof systems, Gentzen-style sequent-calculi, rewrite systems with rule priorities, and the $\pi$-calculus into CERSs. This last encoding is an (important) example of real context-sensitive rewriting.

## 1 Introduction

A term rewriting system is a pair consisting of an alphabet and a set of rewrite rules. The alphabet is used *freely* to generate the terms and the rewrite rules can be applied in any *surroundings*, generating the rewrite relation. In the first order case (no variable binding) one speaks of TRSs while in the higher

[1] Supported by the Engineering and Physical Sciences Research Council of Great Britain under grant GR/H 41300

order case (with variable binding) there exist several conceptually similar, but notationaly often quite different proposals. Long ago, the first general higher order format was introduced by Klop [10] under the name of *Combinatory Reduction Systems*. Since then, several other interesting formalisms have been introduced [7,17,23,19,21]. This paper is based on the notion of *Expression Reduction System* introduced by the first author [7], but our results also apply to the other higher order formats.

Often it is of interest to have the possibility to put restrictions on the generation of either the terms or the rewrite relation (or both). For example, many typed lambda calculi can be viewed as untyped lambda calculus with restricted *term* formation. Let's call them *sub*-ERSs (cf. [12, Def. 12.9]) On the other hand, many rewrite strategies are naturally expressed by restricting application of the *rewrite rules*. For example, the call-by-value strategy in $\lambda$-calculus can be specified by restricting the second *argument* of the $\beta$-rule to values. In general, restricting arguments gives rise to so-called *conditional* ERSs (cf. [5]). The leftmost-outermost strategy can be specified by restricting the *context* in which the $\beta$-rule may be applied. We will call the latter kind of rules in which contexts are restricted *context-sensitive*.[2] In Section 2 we introduce *CERSs* (conditional context-sensitive ERSs) which allow all three kinds of restriction.

In Section 3 we present a suitable notion of orthogonality and prove the standard results for orthogonal CERSs (OCERSs) like the Finite Developments Theorem, confluence etc. by adapting a method for unconditional higher order rewriting [10,7].

In Section 4 we show how some transition and proof systems can be expressed in a natural way in CERSs. A very similar idea is present in the work of Meseguer [14] who encodes many systems in his Conditional Rewriting Logic [14]. Nevertheless, our encoding of calculi with bound variables seems to be more natural, since we don't need to 'code the bindings away' into a first order framework.

## 2 Conditional Expression Reduction Systems

We present CERSs in the style of ERSs [7]. Terms are formed as usual from the alphabet as in the first order case, but for symbols having binding power (like $\lambda$ in $\lambda$-calculus or $\int$ in integrals) which require some binding variables and terms as arguments (as specified by their arity). Scope indicators are used to specify which variables have binding power in which arguments. Note that one cannot substitute for binding variables. The variables for which one can substitute are called metavariables (like in Klop's CRSs).

**Definition 2.1** *Let* $\Sigma$ *be an* alphabet *comprising* variables*, denoted by* $x$*,* $y$*,* $z$ *and* symbols *(*signs*). A symbol* $\sigma$ *can be either a* function symbol *(*simple operator*) having an* arity $n \in N$*, or an* operator sign *(*quantifier sign*) having*

[2] The distinction between 'conditional' and 'context-sensitive' is more a historical than a conceptual one.

arity $(m,n) \in N \times N$. *In the latter case* $\sigma$ *needs to be supplied with* $m$ binding *variables* $x_1,\ldots,x_m$ *to form the* quantifier *(*compound operator*)* $\sigma x_1 \ldots x_m$. *If* $\sigma$ *is an operator sign it also has a* scope indicator *which is a vector of length* $m$ *specifying for each variable in which of the* $n$ *arguments it has binding power.* Terms $t$, $s$, $e$, $o$ *are constructed from variables, function symbols and quantifiers in the usual first order way respecting (the second component of the) arities. A predicate* $AT$ *on terms specifies which terms are* admissible.

Metaterms *are constructed like terms, but also allowing as basic constructions* metavariables $A$, $B$, ... *and* metasubstitutions $(t_1/x_1,\ldots,t_n/x_n)t_0$, *where each* $t_i$ *is an arbitrary metaterm and the* $x_i$ *have binding effect in* $t_0$. *An* assignment *(*substitution*)* $\theta$ *maps each metavariable to some term. The application of the substitution* $\theta$ *to a term* $t$ *is written* $t\theta$ *and is obtained from* $t$ *by replacing metavariables with their values under* $\theta$, *and by replacing metasubstitutions* $(t_1/x_1,\ldots,t_n/x_n)t_0$, *in right to left order, with the result of substitution of terms* $t_1,\ldots,t_n$ *for free occurrences of* $x_1,\ldots,x_n$ *in* $t_0$ *(cf. Kahrs' notion of substitute [12]).*

For example, a $\beta$-redex in the $\lambda$-calculus appears as $Ap(\lambda x\ t,s)$, where $Ap$ is a function symbol of arity 2, and $\lambda$ is an operator sign of arity $(1,1)$ and scope indicator $(1)$. Integrals such as $\int_s^t f(x)\,dx$ can be represented as $\int x(s,t,f(x))$ using an operator sign $\int$ of arity $(1,3)$ and scope indicator $(3)$. The predicate $AT$ can be used to express sorting and typing constraints.

The specification of a CERS consists of a (restricted) alphabet as specified above and a set of (restricted) rules as specified below.

**Definition 2.2** *A rewrite rule is a (named) pair of metaterms* $r : t \to s$, *such that* $t$ *and* $s$ *do not contain free variables. We close the rules under assignments:* $r\theta : t\theta \to s\theta$ *if* $r : t \to s$ *and* $\theta$ *is a substitution. For reasons of hygiene this is restricted to assignments* $\theta$ *such that each free variable occurring in a term* $A\theta$ *assigned to a metavariable* $A$ *is either bound in the* $\theta$*-instance of each occurrence of* $A$ *in the rule or in none of them. The term* $t\theta$ *is then called a* redex *and* $s\theta$ *its* contractum. *Next, we close under contexts* $C[r\theta] : C[t\theta] \to C[s\theta]$, *if* $r\theta : t\theta \to s\theta$ *and* $C[\,]$ *is a context (a term with one hole).*

*The rewrite relation thus obtained is the usual (unconditional, context-free) ERS-rewrite relation. If restrictions are put on assignments, via a predicate* $AA$ *on rules and substitutions, the rewrite relation will be called* conditional. *If restrictions are put on contexts, via a predicate* $AC$ *on rules, substitutions and contexts, the rewrite relation will be called* context-sensitive.

*A* CERS *is a pair consisting of an alphabet and a set of rewrite rules, both possibly restricted.*

In the sequel when we speak about terms and redexes, we will always mean admissible terms and admissible redexes, respectively.

Our syntax is very close to the syntax of the $\lambda$-calculus and of First Order Logic. For example, the $\beta$-rule is written as $Ap(\lambda xA,B) \to (B/x)A$, where $A$ and $B$ can be instantiated by any terms. The $\eta$-rule is written as $\lambda xAp(A,x) \to A$, where it is required that $x \notin A\theta$ for an assignment $\theta$, otherwise an $x$ occurring in $A\theta$ and therefore bound in $\lambda x(A\theta x)$ would become free.

A rule like $f(A) \to \exists x(A)$ is also allowed, but in that case the assignment $\theta$ with $x \in A\theta$ is not. The recursor rule is written as $\mu(\lambda xA) \to (\mu(\lambda xA)/x)A$. Note that we allow metavariable-rules like $\eta^{-1} : A \to \lambda xAp(Ax)$ and metavariable-introduction-rules like $f(A) \to g(A, B)$, which are usually excluded a priori. This is only useful when the system is conditional.

## 3 Orthogonal CERSs

We define orthogonal CERSs (OCERSs) and sketch our proof of Finite Developments for them, implying confluence. The FD proof is based on Nederpelt & Klop's method [16,10] for reducing strong normalization to weak normalization. It is similar in structure to, but simpler than Klop's original confluence proof for orthogonal CRSs [10] and we think not more difficult than other existing confluence proofs [20,17,19,13].

The idea of orthogonality is that contraction of a redex does not destroy others (in whatever way), but rather leaves a number of their residuals. A prerequisite for the definition of residual is the notion of *descendant* allowing to trace subterms during a reduction. Whereas this is simple in the first order case, ERSs may exhibit very complex behaviour due to the possibility of nested metasubstitutions thereby complicating the definition of descendants. Fortunately each rewrite step can be decomposed into two parts: a *TRS*-part replacing the left-hand side by the right-hand side, but without evaluating the metasubstitutions, and a *substitution*-part evaluating the metasubstitutions. This point of view is profitable [3] since the descendant relation of a rewrite step can now be obtained by composing the descendant relation of the TRS-step, which is trivial, and the descendant relations of the evaluation steps, which are a kind of $\beta$-steps (see [7]).

**Definition 3.1** *To an CERS $(\Sigma, R)$ we associate its* refined *version $(\Sigma_{fS}, R_{fS})$, where $\Sigma_{fS}$ is obtained from $\Sigma$ by adding fresh symbols $S^{n+1}$ and $R_{fS}$ is obtained from $R$ by the following procedure*

(i) *Replace each $R$-rule $r : t \to s$ by the rule $r_f : t \to s_f$, where $s_f$ is $s$ with each 'implicit' metasubstitution replaced by its 'explicit' pendant $S$.*

(ii) *Add rules for $S^{n+1}$ (cf. polyadic $\lambda$-calculus [11, p. 115])*

$$S^{n+1}x_1 \dots x_n A_1 \dots A_n A \to (A_1/x_1, \dots, A_n/x_n)A$$

*Obviously, an $r$-step can be simulated by an $r_f$-step followed by a number of $S$-steps. Via the corresponding descendant relations of these steps, this induces a (unique) descendant relation for $r$. Two (admissible) redexes with respect to the same rule are called* weakly similar. *A descendant of a redex $u$ which is a redex weakly similar to $u$ is called a $u$-*residual.

*We call a CERS* orthogonal *(OCERS) if:*

(i) *the left-hand side of a rule is not a single metavariable,*

(ii) *the left-hand side of a rule does not contain metasubstitutions and its*

[3] It even seems to be prerequisite for syntactical studies of higher order rewriting.

*metavariables contain those of the right-hand side,*

(iii) *in no term redex-patterns can overlap,*

(iv) *all the descendants of a redex $u$ in a term $t$ under the contraction of any other redex $v \in t$ are residuals of $u$.*

The second condition ensures that rules exhibit deterministic behaviour when they can be applied. The last condition can be thought of as imposing some closure conditions on arguments and contexts of rules. For example, consider the rules $a \to b$ and $f(A) \to A$ with admissible assignment $\theta A = a$. The descendant $f(b)$ of the redex $f(a)$ after contraction of $a$ is not a redex since the assignment $\theta A = b$ is not admissible, hence the system is not orthogonal (it should not be, since it is not confluent). Note that unconditional non-left-linear rules (almost) never satisfy (iv).

A *development* of a set of non-overlapping redexes is a reduction in which only residuals of redexes in that set are contracted. A development can be conveniently visualized by underlining the head-symbols of the redexes in the set, only allowing contraction of underlined redexes. We will denote the corresponding underlined rewrite system by $\underline{R}$.

**Theorem 3.2** *All developments in an OCERSs $R$ are finite (FD), that is, $\underline{R}$ is strongly normalizing.*

Because space is limited we will contend ourselves with presenting the main ideas of the proof, which follows closely the proof of FD for orthogonal ERSs as presented in [7]. The full proof can be found in the report version [9].

$\underline{R}$ can be refined into $\underline{R}_{fS}$ and surely strong normalisation of the latter implies strong normalisation of the former. To prove strong normalisation of $\underline{R}_{fS}$ the 'memory' technique by Nederpelt and Klop is useful. The idea is to transform the system $\underline{R}_{fS}$ into yet another orthogonal system $\underline{R}^{\mu}_{fS}$ where no erasure takes place, by 'memorizing' metavariables which might be erased. We use a simplified version of Nederpelt & Klop's technique, as developed in [8]. For example

$$\underline{f}(A, B) \to f(A)$$

is transformed into

$$\underline{f}(A, B) \to \mu(B, f(A))$$

where the $B$ is 'memorized' since it did not have descendants in $f(A)$. This $\mu$-transformation is also applied to the $S$-rules. ¿From the definition we immediately have that every $\underline{R}_{fS}$-reduction can be lifted to an $\underline{R}^{\mu}_{fS}$-reduction of the same length, for which the number of $\mu$'s *increases* in each step. Note that the presence of the 'memory' ($\mu$) cannot prevent creation of redexes, since there is no creation of redexes possible in $\underline{R}_{fS}$. Moreover, $\underline{R}^{\mu}_{fS}$ is *weakly normalizing* as can be seen by considering the rightmost-innermost strategy. To conclude strong normalization of $\underline{R}^{\mu}_{fS}$, we can apply the following lemma from [10].

**Lemma 3.3** *A locally confluent, increasing, weakly normalizing abstract rewriting system is strongly normalizing (so confluent by Newman's Lemma).*

¿From the conditions on admissibility in the definition (these conditions

are needed for confluence as witnessed by the example above) of orthogonality and finiteness of developments, one can conclude confluence. Actually, most of Lévy's theory of *permutation equivalence* can be reduced to FD, so is applicable to OCERSs. This is properly addressed in [9].

**Theorem 3.4** *Orthogonal CERSs are confluent.*

Untyped lambda calculus [4] is the prime example of an (unconditional) orthogonal higher-order term rewriting system. If one restricts term formation in it, one arrives at the large class of typed lambda calculi. Since the rewrite relation in these calculi is not restricted in any way, and typed terms are closed under $\beta$-reduction[4] these sub-ERSs are orthogonal, hence confluent.

An interesting example of a CERS was recently studied in [2]. Terms are ordinary $\lambda$-terms possibly containing let expressions, but the rewrite rules have conditions on them as follows. Define the syntactic categories by the following grammar

$$\begin{aligned} M &::= x \mid MM \mid \lambda x.M \mid \mathsf{let}\ x = M\ \mathsf{in}\ M \\ V &::= \lambda x.M \\ A &::= V \mid \mathsf{let}\ x = M\ \mathsf{in}\ A \\ E &::= [\,] \mid EM \mid \mathsf{let}\ x = M\ \mathsf{in}\ E \mid \mathsf{let}\ x = E\ \mathsf{in}\ E[x] \end{aligned}$$

The rules are

$$\begin{aligned} (\lambda x.M)M' &\to \mathsf{let}\ x = M'\ \mathsf{in}\ M \\ \mathsf{let}\ x = V\ \mathsf{in}\ E[x] &\to \mathsf{let}\ x = V\ \mathsf{in}\ E[V] \\ (\mathsf{let}\ x = M\ \mathsf{in}\ A)M' &\to \mathsf{let}\ x = M\ \mathsf{in}\ AM' \\ \mathsf{let}\ x = (\mathsf{let}\ y = M\ \mathsf{in}\ A)\ \mathsf{in}\ E[x] &\to \mathsf{let}\ y = M\ \mathsf{in}\ \mathsf{let}\ x = A\ \mathsf{in}\ E[x] \end{aligned}$$

the rewrite relation $\to_s$ is obtained from this by allowing arbitrary contexts. By some case analysis, one shows that each of the syntactic categories is closed under $\to_s$ and that there are no overlaps between rules, so the system is an orthogonal conditional ERS. Even stronger, the system is leftnormal in the sense of [10], hence standard reductions are normalizing.

An emerging class of context-sensitive and conditional ERSs is the class of $\lambda$-calculi with restricted expansion rules like $\bar{\eta}$ (see e.g. [1]). These calculi are not quite orthogonal, nevertheless their confluence can be shown by tampering with the confluence diagrams arising from FD for the corresponding unconditional expansion rules.

## 4 Expressive power of CERSs

In [14], Meseguer gives encodings of labeled transition systems, several functional programming languages, Chomsky grammars, and some concurrent languages (e.g. Chemical Abstract Machine and CCS), into CTRSs. In this section, we give encodings of some other proof and computation systems, to show that CERSs are even more expressive programming languages. This is not always very useful to understand the original systems better (e.g, one doesn't

[4] This so-called *Subject Reduction* property is sometimes non-trivial to prove.

gain any insight from encoded versions of Hilbert and Gentzen style proof systems), but it often helps to understand operational behaviour of a system (e.g., in the case of the $\pi$-calculus).

### *4.1 Conditional TRSs*

Conditional term rewriting systems (CTRSs) were introduced by Bergstra & Klop in [5]. Their conditional rules have the form $t_1 = s_1 \wedge \cdots \wedge t_n = s_n \Rightarrow t \to s$, where the $s_i$ and $t_i$ may contain variables in $t$ and $s$. According to such a rule $t\theta$ can be rewritten to $s\theta$ if all the equations $s_i\theta = t_i\theta$ are satisfied. CTRSs were classified depending on how satisfaction is defined ('=' can be interpreted as $\twoheadrightarrow$ , $\leftrightarrow^*$, etc.) As they remark this can be generalized by allowing for arbitrary predicates on the variables as conditions (cf. also [6,22]).

Clearly, all these CTRSs are context-free CERSs since they only allow conditions on the arguments not on the context of rewrite rules. For this reason sometimes results for them are a special case of general results which hold for all CERSs. In particular, *stable* CTRSs for which the unconditional version is orthogonal as defined in [5] are orthogonal in our sense, so confluent. Several confluence results were obtained in the above papers for non-orthogonal CTRSs as well, which perhaps can also be generalized to the higher-order case.

### *4.2 Encoding of strategies*

In the literature ([4]) a strategy for a rewriting system $(R, \Sigma)$ is a map $F{:}Ter(\Sigma) \to Ter(\Sigma)$, such that $t \to F(t)$ if $t$ is not a normal form, and $t = F(t)$ otherwise. Such strategies are deterministic and only specify what to do, not how to do it. We prefer to view a strategy as a set $F$ of triples $(r, \theta, C[\,])$ specifying that rule $r : t \to s \in R$ can be used with assignment $\theta$ in context $C[\,]$ to rewrite $C[t\theta]$ to $C[s\theta]$.[5] To a strategy $F$ one can associate a CERS $R_F$ encoding exactly the same information, by taking $\theta, C[\,]$ admissible for $r$ iff $(r, \theta, C[\,]) \in F$. Obviously, this also holds the other way around, that is, every CERS can be viewed as a strategy for its unconditional version.

### *4.3 Encoding of rewrite systems with priorities*

A priority rewrite system, or PRS for short is a pair consisting of a TRS $R$ and a partial order $<$ on the set of rules of $R$ [3]. The partial order is meant as a judge in case of conflict between rules. An $r$-redex $u$ can be contracted iff it is a closed term, and there is no $r' > r$ such that $u$ can be rewritten to an $r'$-redex by means of an internal (i.e. taking place below the headsymbol) reduction; such redexes ar allowed to be contracted in any context. Because of the negative condition in the definition of the rewrite relation, PRSs are not always well-defined, but it is clear that those which are well-defined can

[5] Note that an ordinary strategy $F$ can be directly encoded by associating the set $\{(r : t \to s, \theta, C[\,]) \mid r \in R, C[s\theta] = F(C[t\theta])\}$ to it.

be expressed as a conditional ERS. In [3] some criteria sufficient for well-definedness as well as for ground confluence are found. In particular, it is shown that *essentially regular*[6] RPSs are ground confluent. Such PRSs are orthogonal in our sense, so this confluence result is covered by ours.

### *4.4 Encoding of Hilbert style proof systems*

To illustrate the expressive power of CERSs we give an encoding of Hilbert style proof systems into CERSs, translating deduction into reduction. A Hilbert style system $H$ has a number of axioms $F_1, F_2, \ldots$ and two rules: *Modus Ponens*, allowing to infer $B$ when $A$ and $A \Rightarrow B$ are theorems, and the $\exists$*-rule*, allowing to infer $\exists x.A[x]$ if $A[t]$ is a theorem. A proof in the axiomatic theory $H$ is a finite sequence of formulae $G_1, G_2, G_3, \ldots, G_m$ such that $G_i$ is either an axiom (i.e., coincides with one of the $F_j$) or is obtained from $G_1, \ldots, G_{i-1}$ by one of the above two rules. To each $H$ we can associate a CERS $R_H$ as follows. The alphabet of $R_H$ consists of the alphabet of $R$ augmented by the function symbols $P^n$ of arity $n$, used to model the current stock of theoremata. The rules, more precisely the rule-schemata, of $R_H$ are:

- $P^n(A_1, \ldots, A_n) \to P^{n+1}(A_1, \ldots, A_n, F)$, for each $n$ and axiom $F$. In particular $P^0 \to P^1(F)$. Admissible assignments assign arbitrary formulae to the metavariables $A_1, \ldots, A_n$, and an axiom to the metavariable $F$.
- $P^n(A_1, \ldots, A_k, \ldots, A_k \Rightarrow A, \ldots, A_n) \to P^{n+1}(A_1, \ldots, A_n, A)$ for each $n \geq 2$. The $A_k$ may also appear after $A_k \Rightarrow A$ in the sequence. Admissible substitutions are the same as in the previous case.
- $P^n(A_1, \ldots, (A/x)A_k, \ldots, A_n) \to P^{n+1}(A_1, \ldots, (A/x)A_k, \ldots, A_n, \exists a A_k)$ for each $n \geq 1$. An admissible assignment $\theta$ assigns arbitrary formulae to $A_1, \ldots, A_n$ and a term to $A$.

Obviously there is a 1–1 correspondence between theoremata of $H$ and terms which occur as argument of some $P^n$ in a $R_H$-reduction starting from $P^0$.

Encoding a Gentzen style proof system is similar to a Hilbert style system, the main idea being to translate inference rules into rewrite rules, proofs into terms and deduction into reduction. We refer to [9] for full treatement.

### *4.5 Encoding of the $\pi$-calculus*

In this paragraph we will encode the version of $\pi$-calculus as described in [15] as a CERS. Recall that $\pi$-calculus agents $P$, $Q$, ... are defined as follows:

$$P ::= \overline{x}y.P \mid x(y).P \mid 0 \mid P|P \mid !P \mid (x)P$$

Basic interaction is generated from the rule

$$x(y).P|\overline{x}z.Q \to [z/y]P|Q$$

by closing under unguarded contexts and working modulo structural congruence (see [15]).

[6] The left-linearity condition in [3] is redundant, since it is implied by essential nonambiguity.

To $\pi$-calculus a CERS $(\Sigma_\pi, R_\pi)$ can be associated as follows. The alphabet $\Sigma_\pi$ consists of the function symbols $0, !, |, O$ with respective arities $0, 1, 2, 3$, and the quantifier symbols $I$ and $R$ with arities $(1,2)$ and $(1,1)$. $I$ binds only in its last argument. The map $[\,]$ transforms $\pi$-terms into terms in $Ter(\Sigma_\pi)$. The only non-obvious cases are input, output and restriction:

$$[x(y).P] = Iy(x,[P])\,;\,[\overline{x}z.Q] = O(x,z,[Q])\,;\,[(x)P] = Rx([P])$$

Combining the transformation $[\,]$ with the closing under unguarded contexts and the structural congruence leads to rules $R_\pi$ of the form

$$C_1[Iy(X,P)] \,|\, C_2[O(X,Z,Q)] \to C_1[(Z/y)P] \,|\, C_2[Q], \text{ where}$$

(i) $P, Q, X, Z$ are metavariables, and admissible assignments for $X, Z$ are variables.

(ii) The indicated subterms must be unguarded in $C_1[\,]$ and $C_2[\,]$ and not in the scope of $RX$ (among the symbols above them only the operators $|$, $!$ and $Rx$ with $x \neq X$ can occur).

(iii) Only (all) unguarded contexts are admissible, for any redex.

Obviously, the 'critical pairs' for the interaction rule are preserved by the translation, so $R_\pi$ is not orthogonal. Nevertheless, we expect results like: for the standard translation of $\lambda$- into $\pi$-calculus, $R_\pi$ is orthogonal hence confluent modulo the structural congruence.

## References


[1] Akama Y. On Mints' reduction for ccc-calculus. In: Proc. of the $1^{st}$ International conference on Typed Lambda Calculus and Applications, TLCA'93, LNCS, vol. 664, Bezem M., Groote J.F., eds., 1993, p. 1–12.

[2] Ariola Z.M., Felleisen M., Maraist J., Odersky M., Wadler P. A Call-By-Need Lambda Calculus. In: Proc. ACM Conference on Principles of Programming Languages, 1995.

[3] Baeten J.C.M., Bergstra J.A., Klop J.W., Weijland W.P. Term Rewriting Systems with rule priorities. Journal of TCS 67, 1989, p. 283–301.

[4] Barendregt H.P. The Lambda Calculus, its Syntax and Semantics. North-Holland, 1984.

[5] Bergstra J. A., Klop J. W. Conditional Rewrite Rules: confluence and termination. JCSS vol. 32, n. 3, 1986, p. 323–362.

[6] Dershowitz N., Okada M., Sivakumar G. Canonical conditional rewrite systems. In: Proc. of the $9^{th}$ International Conference on Automated Deduction, LNCS, vol. 310, p. 538–549.

[7] Khasidashvili Z. The Church-Rosser theorem in Orthogonal Combinatory Reduction Systems. Report 1825, INRIA Rocquencourt, 1992.

[8] Khasidashvili Z. The longest perpetual reductions in orthogonal expression reduction systems. In: Proc. of the $3^{rd}$ International Conference on Logical

Foundations of Computer Science, LFCS'94, LNCS, vol. 813, Nerode A., Matiyasevich Yu. V. eds. St. Petersburg, 1994. p. 191–203.

[9] Khasidashvili Z., van Oostrom V. Context-sensitive Conditional Rewrite Systems. Report SYS-C95-06. University of East Anglia, 1995.

[10] Klop J.W. Combinatory Reduction Systems. Mathematical Centre Tracts 127. Centre for Mathematics and Computer Science, Amsterdam, 1980.

[11] Klop J.W. Term Rewriting Systems. Report CS-R9073, Centre for Mathematics and Computer Science, 1990.

[12] Klop J. W., van Oostrom V., van Raamsdonk F. Combinatory reduction systems: introduction and survey. In: To Corrado Böhm, J. of Theoretical Computer Science 121, 1993, p. 279–308.

[13] Melliès P.-A. An abstract theorem of finite developments. Talk presented at CONFER-meeting, september 1993, Amsterdam.

[14] Meseguer J. Conditional Rewriting Logic as a unified model of concurrency. Journal of TCS 96, 1992, p. 73–155.

[15] Milner R. Functions as processes. In: Journal of Mathematical structures in Computer Science. 2(2): 1992, p. 119–141.

[16] Nederpelt R.P. Strong normalization for a typed lambda-calculus with lambda structured types. Ph.D. Thesis, Eindhoven, 1973.

[17] Nipkow T. Orthogonal higher-order rewrite systems are confluent. In: Proc. of the $1^{st}$ International conference on Typed Lambda Calculus and Applications, TLCA'93, LNCS, vol. 664, Bezem M., Groote J.F., eds., 1993, p. 306-317.

[18] Van Oostrom V. Confluence for Abstract and Higher-Order Rewriting. Ph. D. Thesis, Free University of Amsterdam, 1994.

[19] Van Oostrom V., van Raamsdonk F. Weak orthogonality implies confluence: the higher-order case. In: Proc. of the $3^{rd}$ International Symposium on Logical Foundations of Computer Science, LFCS'94, LNCS, vol. 813, Nerode A., Matiyasevich Yu. V., eds. St. Petersburg, 1994, p. 379-392.

[20] Van Raamsdonk F. Confluence and superdevelopments. In: Proc. of the $5^{th}$ International Conference on Rewriting Techniques and Applications, RTA'93, LNCS, vol. 690, C. Kirchner, ed. Montreal, 1993, p. 168-182.

[21] Takahashi M. $\lambda$-Calculi with Conditional Rules. In: Proc. of the $1^{st}$ International conference on Typed Lambda Calculus and Applications, TLCA'93, LNCS, vol. 664, Bezem M., Groote J.F., eds., 1993, p. 406–417.

[22] Toyama Y. Confluent term rewriting systems with membership conditions. In: Proc. of the $1^{st}$ International Workshop on Conditional Term Rewriting Systems, LNCS, vol. 308, 1988, p. 128–141.

[23] Wolfram D. The clausal theory of types. Cambridge Tracts in Theoretical Computer Science, vol. 21, Cambridge University Press, 1993.