Context-sensitive Conditional Reduction Systems

Z. Khasidashvili and V. van Oostrom

Technical Report SYS-C95-06

UEA Norwich, UK

Context-sensitive Conditional Reduction Systems

Zurab Khasidashvili [1]
zurab@sys.uea.ac.uk
School of Information Systems
University of East Anglia
Norwich NR4 7TJ, UK

Vincent van Oostrom[2]
oostrom@cs.vu.nl
Department of Mathematics and Computer Science
Vrije Universiteit, De Boelelaan 1081a
1081 HV Amsterdam, The Netherlands

Technical Report SYS-C95-06

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 λ-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 π-calculus into CERSs. This last encoding is an important example of real context-sensitive rewriting.

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

[2]Work partially performed while at: NTT Basic Research Laboratories, Information Processing Principles Research Group, 3–1 Morinosato Wakamiya, Atsugi-shi, Kanagawa-ken, 243-01, Japan

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* (context), generating the rewrite relation. In the first order case (no variable binding) one speaks of TRSs while in the higher 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 [Klo80] under the name of *Combinatory Reduction Systems*. Since then, several other interesting formalisms have been introduced [Kha92, Nip93, Wol93, OR94, Tak93, Lor93]. This paper is based on the notion of *Expression Reduction System* introduced by the first author [Kha92], 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. [KOR93, 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 λ-calculus can be specified by restricting the second *argument* of the β-rule to values. In general, restricting arguments gives rise to so-called *conditional* ERSs (cf. [BK86]). The leftmost-outermost strategy can be specified by restricting the *context* in which the β-rule may be applied. We will call the latter kind of rules in which contexts are restricted *context-sensitive*.[3] 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 [Klo80, Kha92].

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 [Mes92] who encodes many systems in his Conditional Rewriting Logic [Mes92]. 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 [Kha92]. Terms are formed as usual from the alphabet as in the first order case, but for symbols having binding power (like λ in λ-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 Σ be an* alphabet *comprising* variables*, denoted by x, y, z and* symbols *(*signs*). A symbol σ can be either a* function symbol *(*simple operator*) having an* arity *$n \in N$,*

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

or an operator sign *(*quantifier sign*) having* arity $(m, n) \in N \times N$. *In the latter case* σ *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* σ *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. *Metaterms without metasubstitutions are called* simple. *An* assignment *(*substitution*)* θ *maps each metavariable to some term. The application of the substitution* θ *to a term* t *is written* $t\theta$ *and is obtained from* t *by replacing metavariables with their values under* θ, *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 [KOR93]).*

For example, a β-redex in the λ-calculus appears as $Ap(\lambda x\ t, s)$, where Ap is a function symbol of arity 2, and λ 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 an alphabet (generating a set of terms possibly restricted by the predicate AT) as specified above and a set of rules (generating the rewrite relation possibly restricted by the predicates AA and AC) as specified below. *Sub*-ERSs (short for substructure ERSs) in the same sense as Klop's sub-CRSs (cf. [KOR93]) are CERSs where the predicate AT is non-trivial.

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* θ *is a substitution. For reasons of hygiene this is restricted to assignments* θ *such that*

(a) each free variable occurring in a term $A\theta$ *assigned to a metavariable* A *is either bound in the* θ*-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 an admissibility *predicate* AA *on rules and assignments, the rewrite relation will be called* conditional. *We call redexes that are instances of the same rule (i.e., with the same admissibility predicate)* weakly similar. *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 R *is a pair consisting of an alphabet and a set of rewrite rules, both possibly restricted.* R *is* simple *if right-hand sides of all its rules are simple metaterms.*

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 λ-calculus and of First Order Logic. For example, the β-rule is written as $Ap(\lambda xA, B) \to (B/x)A$, where A and B can be instantiated by any terms. The η-rule is written as $\lambda xAp(A,x) \to A$, where it is required that $x \notin A\theta$ for an assignment θ, 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 θ 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 [Ned73, Klo80] for reducing strong normalization to weak normalization. It is similar in structure to, but simpler than Klop's original confluence proof for orthogonal CRSs [Klo80] and we think not more difficult than other existing confluence proofs [vR93, Nip93, OR94, Mel93].

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[4] 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 β-steps (see [Kha92]).

To express substitution, we use the S-reduction rules

$$S^{n+1}x_1 \dots x_n A_1 \dots A_n A_0 \to (A_1/x_1, \dots, A_n/x_n)A_0, \;\; n = 1, 2, \dots,$$

where S^{n+1} is the *operator sign of substitution* with arity $(n, n+1)$ and scope indicator $(n+1)$, and $x_1, \dots, x_n$ and $A_1, \dots, A_n, A_0$ are pairwise distinct variables and metavariables. Thus S^{n+1} binds free variables only in the last argument. The difference with β-rules is that S-reductions can only perform β-developments of λ-terms [Kha92], so one can think of them as (simultaneous) let-expressions.

Notation 3.1 *We use a, b, c, d for constants, t, s, e, o for terms and metaterms, u, v, w for redexes, and N, P, Q for reductions. We write $s \subseteq t$ if s is a subterm of t. A one-step reduction in which a redex $u \in t$ is contracted is written as $t \xrightarrow{u} s$ or $t \to s$ or just u. We write $P : t \twoheadrightarrow s$ if P denotes a reduction of t to s. $|P|$ denotes the length of P. $P + Q$ denotes the concatenation of P and Q.*

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

We now recall briefly but precisely the definition of *descendant* of subterms as introduced in [Kha88, Kha93, Kha92] for the λ-calculus, TRSs, and ERSs, respectively. First, we split an ERS R into a *TRS-part* R_f and the *substitution-part* S. For any ERS R, which we assume does not contain symbols S^{n+1}, R_f is the ERS obtained from R by adding symbols S^{n+1} in the alphabet and by replacing in right-hand sides of the rules all metasubstitutions of the form $(t_1/x_1, \ldots, t_n/x_n)t_0$ by $S^{n+1}x_1 \ldots x_n t_1 \ldots t_n t_0$, respectively. For example, the β_f rule would be $\beta_f : Ap(\lambda xA, B) \to S^2 xBA$. If R is simple, then $R_{fS} =_{def} R_f =_{def} R$. Otherwise $R_{fS} =_{def} R_f \cup S$. For each step $C[t\theta] \xrightarrow{u} C[s\theta]$ in R there is a reduction $P : C[t\theta] \to_{R_f} C[s'\theta] \twoheadrightarrow_S C[s\theta]$ in R_{fS}, where $C[s'\theta] \twoheadrightarrow_S C[s\theta]$ is the rightmost innermost normalizing S-reduction. We call P the *refinement* of u. For example, the refinement of the β-step $Ap(\lambda xt, s) \to_\beta (s/x)t$ would be $Ap(\lambda xt, s) \to_{\beta_f} S^2 xst \to_S (s/x)t$. The notion of refinement generalizes to R-reductions with 0 or more steps.

Let $t \xrightarrow{u} s$ be an R_f-reduction step and let e be the contractum of u in s. For each argument o of u there are zero or more arguments of e. We call them (u-)*descendants* of o. Correspondingly, subterms of o have zero or more *descendants*. The *descendant* of each pattern-subterm (i.e., a subterm rooted within the pattern) of u is e. It is clear what is to be meant by *descendants* of a subterm that is not in u. For example, in the step $t = f(g(a)) \to b = s$ according to the rule $f(g(x)) \to b$, the descendant of both pattern-subterms $f(g(a))$ and $g(a)$ of t in s is b, and a doesn't have a descendant in s.

In an S-reduction step $C[S^{n+1}x_1 \ldots x_n t_1 \ldots t_n t_0] \xrightarrow{u} C[(t_1/x_1, \ldots, t_n/x_n)t_0]$, the argument t_i and subterms in t_i have the same number of *descendants* as the number of free occurrences of x_i in t_0. All subterms of t_0 have exactly one *descendant* (the descendants of the free occurrences of $x_1, \ldots, x_n$ in t_0 are the substituted subterms). The *descendant* of the contracted redex u itself is its contractum. For example, in the step $o = Sxaf(x) \to f(a) = e$, the descendant of $a \in o$, as well as of the bound occurrence of x in $f(x)$, is the occurrence of a in e. The descendant of $f(x)$, as well as of o itself, is e. The notion of *descendant* extends by transitivity to arbitrary R_{fS}-reductions. If P is an R-reduction, then *P-descendants* are defined to be the descendants under the refinement of P. The *ancestor* relation is converse to that of descendant.

Remark 3.1 *In the literature several slightly different notions of descendants appear, e.g. in [Klo80] the contracted redex is defined not to have any descendants, while in our definition it has exactly one descendant.*

Co-initial reductions $P : t \twoheadrightarrow s$ and $Q : t \twoheadrightarrow e$ are called *strictly equivalent* (written $P \approx_{st} Q$) if $s = e$ and P-descendants and Q-descendants of any subterm of t are the same in s and e [Kha88]; P and Q are *equivalent* (written $P \approx Q$) if $s = e$.

We recall that A CERS R is *weakly Church-Rosser* (WCR) if for any two co-initial (admissible) steps $t \to s$ and $t \to o$ in R, s and o have a common reduct. R is *confluent* or equivalently *Church-Rosser* (resp. *strictly Church-Rosser*) if, for any co-initial reductions $P : t \twoheadrightarrow s$ and $Q : t \twoheadrightarrow e$, there are reductions $P' : s \twoheadrightarrow o$ and $Q' : e \twoheadrightarrow o$ such that $P + P' \approx Q + Q'$ (resp. $P + P' \approx_{st} Q + Q'$).

Definition 3.1 *Let $t \xrightarrow{u} s$ in an OCERS R, let $v \in t$ be an admissible redex, and let $w \in s$ be a u-descendant of v. We call w a u-*residual *of v if (a) the patterns of u and v do not overlap; (b) w is a redex weakly similar to v; and (c) w is admissible. (So u itself does not have u-*residuals *in s.) The notion of* residual *of redexes extends naturally to arbitrary reductions. A redex in s is called a* new *redex or a* created *redex if it is not a residual of a redex in t.*

Definition 3.2 *We call a CERS* orthogonal *(OCERS) if:*

1. *the left-hand side of a rule is not a single metavariable,*
2. *the left-hand side of a rule does not contain metasubstitutions and its metavariables contain those of the right-hand side,*
3. *in no term admissible redex-patterns can overlap,*
4. *all the descendants of an admissible redex u in a term t under the contraction of any other admissible 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 (4).

We now present the main ideas of the proof of finiteness of developments in an OERS R. Recall that a *development* of a set of non-overlapping (admissible) redexes F in t (notation $F \in t$) or an F*-development* is a reduction $P : t \twoheadrightarrow s$ in which only residuals of redexes in F are contracted; P is a *complete* F-development if s doesn't contain residuals of redexes in F. A development in R can be conveniently visualized by underlining the head-symbols of the redexes in the set, only allowing contraction of underlined redexes. We denote the corresponding underlined rewrite system by $\underline{R}$ (i.e., $\underline{R}$-redexes are R-redexes with the head-symbols of left-hand sides underlined). *Admissible* terms in $\underline{R}$ are those which may contain underlined symbols only as head-symbols of redexes. Thus, termination of developments in R is equivalent to strong normalization (SN) of $\underline{R}$.

$\underline{R}$ 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 [Kha94]. 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 μ-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 μ's *increases* in each step, i.e., $\underline{R}^{\mu}_{fS}$ is *increasing* in the sense of [Klo80]. Note that the presence of the 'memory' (μ) cannot prevent creation of redexes, since there is no creation of redexes possible in $\underline{R}_{fS}$. Therefore, the application of the techniques from [Kha94] is even simpler in this case. Moreover, $\underline{R}^{\mu}_{fS}$ is *weakly normalizing* as can be seen by considering the rightmost-innermost strategy. Strong normalisation of $\underline{R}^{\mu}_{fS}$ now follows from the following lemma of Klop [Klo80].

Lemma 3.1 *[Klo80] A locally confluent, increasing, weakly normalizing abstract rewriting system is strongly normalizing (so confluent by Newman's Lemma).*

We now make the above said more precise.

Notation If $F \in t$ and $P : t \twoheadrightarrow s$, then F/P denotes the set of all residuals of redexes from F in s. We write u/P for $\{u\}/P$. If $F \in t$, then F will also denote the leftmost innermost complete F-development.

Lemma 3.2 *Let t_0 be a term in an OERS R and $F \in t_0$ be a set of redexes in t_0. Then for any F-development $P : t_0 \stackrel{u_0}{\to} t_1 \stackrel{u_1}{\to} \ldots$ there is a reduction $Q : s_0 \stackrel{v_0}{\to} s_1 \stackrel{v_1}{\to} \ldots$ in $\underline{R}$ such that s_i is obtained from t_i by underlining head symbols of redexes from $F_i = F/(u_0 + \ldots u_{i-1})$, and u_i and v_i are corresponding redexes in t_i and s_i for all $i = 0, 1, \ldots$*

Proof Easy.

Definition 3.3 *Let R be an OERS, let $\underline{\Sigma}_{fS}$ be the alphabet of $\underline{R}_{fS}$, and let $\underline{\Sigma}^{\mu}_{fS} = \underline{\Sigma}_{fS} \cup \{\mu^n \mid n = 0, 1, \ldots\}$, where μ^n is a fresh simple operator with arity n (we shall omit n in μ^n). By definition, the μ-rules have the form $\Im = \{\mu^n(A_1 \ldots A_n) \to A_n \mid n = 1, 2, \ldots\}$. We denote the $\Im$-nf of an $\underline{\Sigma}^{\mu}_{fS}$-term t by $[t]_\mu$ ($\Im$ is WCR and strongly normalising (SN), hence is CR by Newman's Lemma). Let $r : t \to e$ be a rule in $\underline{R}_f$ and let $\theta \in AA(r)$. Then, for any assignment θ_μ that satisfies the condition (a) of Definition 2.2, $t\theta_\mu$ is an $\underline{R}^{\mu}_{f}$-*redex. *An $\underline{R}^{\mu}_{f}$-*reduction *step is a replacement of an $\underline{R}^{\mu}_{f}$-*redex *$v = t\theta_\mu = C[s_1, \ldots, s_m]$ (with the pattern $C[\ ]$ and arguments $s_1, \ldots, s_m$) by $\mu s_{j_1} \ldots s_{j_l} e\theta_\mu$, where $s_{j_1}, \ldots, s_{j_l}$ are arguments of v that do not have descendants in s. S^μ-redexes have the form $u = Sx_1 \ldots x_n t_1 \ldots t_n t_0$, where t_i are arbitrary terms over $\underline{\Sigma}^{\mu}_{fS}$. By definition, a step of S^μ-*reduction *is a replacement of an S^μ-*redex *u by $\mu t_{i_1} \ldots t_{i_k} t_0^*$, where $t_0^* = (t_1/x_1, \ldots, t_n/x_n)t_0$, and $t_{i_1}, \ldots, t_{i_k}$ are all arguments of u that do not have descendants in t_0^* (i.e., $x_{i_j} \notin FV(t_0)$ for $j = 1, \ldots, k$).* Admissible terms *over $\underline{\Sigma}^{\mu}_{fS}$ are such $\underline{\Sigma}^{\mu}_{fS}$-terms that underlined symbols may appear only as head-symbols of $\underline{R}^{\mu}_{fS}$-redexes. An $\underline{R}^{\mu}_{fS}$-*reduction *step starting from an admissible term over $\underline{\Sigma}^{\mu}_{fS}$ is an S^μ-reduction step or an $\underline{R}^{\mu}_{f}$-reduction step in any context (i.e., all contexts are admissible). The notions of* descendant, residual, development*, etc. are defined analogously for $\underline{R}^{\mu}_{fS}$-reductions.*

Lemma 3.3 *Let R be an OERS. Then $\underline{R}^{\mu}_{fS}$ is strongly normalizing and Church-Rosser.*

Proof It is routine to check that $\underline{R}^{\mu}_{fS}$ is weakly Church-Rosser. Further, $\underline{R}^{\mu}_{fS}$ is weakly normalizing because clearly innermost $\underline{R}^{\mu}_{fS}$-reductions are normalizing since there is no creation of redexes in $\underline{R}^{\mu}_{fS}$. To conclude strong normalization and confluence of $\underline{R}^{\mu}_{fS}$, we can apply Lemma 3.1.

Lemma 3.4 *Let R be an OERS and $P : t_0 \to t_1 \to \ldots$ be a reduction in $\underline{R}_{fS}$. Then there is a reduction $Q : s_0 = t_0 \to s_1 \to \ldots$ in $\underline{R}^{\mu}_{fS}$ such that $[s_i]_\mu = t_i$ for all $i = 0, 1, \ldots$*

Proof It is routine to check that μ-steps commute with $\underline{R}^{\mu}_{fS}$-steps, and the lemma follows.

Lemma 3.5 *Let R be an OERS. Then $\underline{R}_{fS}$ is strongly normalizing and Church-Rosser.*

Proof Strong normalisation for $\underline{R}_{fS}$ follows from Lemmas 3.3 and 3.4. Confluence follows by Newman's Lemma, since it is easy to check that $\underline{R}_{fS}$ is weakly Church-Rosser.

Theorem 3.1 *Let R be an OERS. Then all developments in R are finite and all complete co-initial developments end at the same term.*

Proof From Lemmas 3.2 and 3.5.

Let $F = \{u_1, \ldots, u_n\}$ be a set of admissible redexes of an admissible term t_0, in an OCERS R. Let r_i be the R-rule for u_i, and let r'_i be the rule obtained from r_i by equipping each function or a quantifier symbol in the LHS by the superscript i. Obviously, $R' = \{r'_i\}$ is an orthogonal ERS; we assume that in R' all terms constructed using symbols from R and the indexed symbols are admissible, that all substitutions are admissible for all R'-rules, and that all contexts are admissible for rewriting. Let G associate to an R'-term the corresponding term in R without indexed symbols (they are admissible). Let s_0 be obtained from t_0 by equipping every symbol in the pattern of u_i with the superscript i, for all i. Then for any F-development $t_0 \to t_1 \to \ldots \to t_n$ in R there is a development $s_0 \to s_1 \to \ldots \to s_n$ of s_0 in R' such that $G(s_j) = t_j$. Therefore:

Theorem 3.2 (Finite Developments) *All developments in an OCERS are finite and all complete co-initial developments end at the same term.*

As a corollary, we get that for any co-initial developments P and Q in an OCERS, there are developments P' and Q' such that $P + P' \approx Q + Q'$, and confluence for OCERSs follows.

Theorem 3.3 (Church-Rosser) *Orthogonal conditional ERSs are confluent.*

Remark 3.2 *The above proof of FD for OERSs follows closely the one in [Kha92], but is simpler. The difference is that, since $\underline{R}^{\mu}_{fS}$ is increasing, here we use use Klop's lemma to conclude SN for $\underline{R}^{\mu}_{fS}$ from WN, while in [Kha92] we prove that every term in $\underline{R}^{\mu}_{fS}$ has exactly one normal form, and SN follows from WN immediately (without using Klop's lemma).*

Since the Parallel Moves Lemma is a corollary of the Finite Developments theorem, one can in a standard way (e.g., as in [HL91, Bar84]) define, for any co-initial reductions P and Q, the notion of the *residual* P/Q of P under Q and prove the strong version of CR for OCERSs.

Theorem 3.4 (Strong Church-Rosser) *For any co-initial reductions P and Q in an OCERS, $Q + P/Q \approx_L P + Q/P$.*

Using finiteness of developments, it is shown in [Kha92] that all co-initial complete developments in OERSs are strictly equivalent (by showing that co-initial reduction steps 'strictly commute'). Clearly, this is valid for OCERSs too, and we can extend the *strict* CR theorem [Kha92] from OERSs to OCERSs.

Theorem 3.5 (Strict Church-Rosser) *For any co-initial reductions P and Q in an OCERS, $Q + P/Q \approx_{st} P + Q/P$.*

Untyped lambda calculus [Bar84] 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 β-reduction[5] these sub-ERSs are orthogonal, hence confluent.

An emerging class of context-sensitive and conditional ERSs is the class of λ-calculi with restricted expansion rules like $\bar{\eta}$ (see e.g. [Aka93]). 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.

We conclude this section with a somewhat larger example of an OCERS.

3.1 Call-by-need λ-calculus

We will show that the *call-by-need λ-calculus* as introduced and studied in [AFMOW94] is an OCERS. Terms in this calculus are ordinary λ-terms possibly containing let expressions, but the rewrite rules have conditions on them as follows. Define the syntactic categories by the following grammar

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

The rules are

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

the rewrite relation $\to_s$ is obtained from this by allowing arbitrary contexts.

By case analysis we show 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.

- M is obviously $\to_s$-closed and contains V, A, and $E[y]$ for every y.
- V is $\to_s$-closed by the previous item and the fact that no root-steps are possible.
- A is $\to_s$-closed since V is so by the previous item and $\mathsf{let}\ x = M\ \mathsf{in}\ A$ is by considering (root-)overlaps with the four rewrite rules.
    1. Root-overlap with the first or third rule is syntactically not possible.
    2. To show that root-overlap with the second rule and fourth rule is not possible it suffices to show that no element in A is in $E[y]$ for any y, which we prove by induction on the definition of A:

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

(a) $V \notin E[y]$ since $E[y] \not\equiv \lambda x.M$.
(b) (let $x = M$ in $A) \notin E[y]$ since
i. (let $x = M$ in $A) \not\equiv y$,
ii. (let $x = M$ in $A) \not\equiv E[y]N$,
iii. (let $x = M$ in $A) \not\equiv$ (let $z = N$ in $E[y]$) by induction hypothesis, and
iv. (let $x = M$ in $A) \not\equiv$ (let $z = E[y]$ in $E[z]$) by induction hypothesis.

- $E[y]$ is shown to be $\rightarrow_s$-closed by induction on the definition of E

1. y is a normal form.
2. $E[y]M$ cannot be root-rewritten since $E[y] \not\equiv \lambda x.N$. $E[y]$ and M are $\rightarrow_s$-closed by hypothesis.
3. let $x = M$ in $E[y]$ cannot be root-rewritten ($x \not\equiv y$!), and M and $E[y]$ are $\rightarrow_s$-closed by hypothesis.
4. let $x = E[y]$ in $F[x]$ cannot be root-rewritten since V and $E[y]$ are disjoint (second rule), and A and $G[z]$ are disjoint (fourth rule). Both $E[y]$ and $F[x]$ are $\rightarrow_s$-closed by hypothesis.

Because of $\rightarrow_s$-closedness of the syntactic categories, to show orthogonality we need only to check for possible 'critical pairs' between the rules. These cannot occur as one easily verifies by using the earlier observation that elements in A are not in $E[y]$. Even stronger, the system is left-normal in the sense of [Klo80], hence standard reductions are normalizing.

4 Expressive power of CERSs

In [Mes92], 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 gain any insight from encoded versions of Hilbert and Gentzen style proof systems), but it often helps to understand the operational behaviour of a system (e.g., in the case of the π-calculus).

4.1 Conditional TRSs

Conditional term rewriting systems (CTRSs) were introduced by Bergstra & Klop in [BK86]. Their conditional rules have the form $t_1 = s_1 \wedge \cdots \wedge t_n = s_n \Rightarrow t \rightarrow s$, 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 [DOS88, Toy88]).

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 [BK86] 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.

More recently, the higher-order generalisation of (join) CTRSs was introduced by Loría-Sáenz under the name of HCTRSs [Lor93]. Like in the first-order case these are context-free and fit in our definition. The confluence results obtained by him are for strongly normalising systems, and are incomparable to ours.

4.2 Encoding of strategies

Given an ERS R and a strategy F for it, we will construct a CERS R_F where exactly the R-reductions performed according to the strategy F are possible. To be more precise, we only consider history-free strategies, i.e., the step chosen just depends on the term, not on the rewrite sequence leading to it.

In the literature a strategy 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. For our purposes the above definition of strategies is not satisfactory. We need a strategy not just to specify what term to reach next, but also how to do this.

First, in a term there may be disjoint redex occurrences yielding the same result if reduced, e.g. take simply the TRS $R = \{f(x) \to a,\ b \to b\}$ and the term $t = g(b, f(b))$. Then t rewrites to itself by rewriting either the first or the second occurrence of b in it. The leftmost b is needed [HL91], while the rightmost is not. Here the information that a strategy F rewrites t to t is not enough to know whether F rewrites a needed redex in t or a non-needed one. So a strategy should specify which redex-occurrence must be contracted.

Second, a redex-occurrence can be an instance of more than one rule, i.e. $LHS(r_1)\theta_1 = u = LHS(r_2)\theta_2$ for some rules r_1 and r_2 and some assignments $\theta_1 \in AA(r_1)$ and $\theta_2 \in AA(r_2)$. The contracta of the redexes can be the same (for example in the parallel or rule), which shows that even knowing the occurrence of the redex does not suffice to know which rule has been applied (unless the context was only admissible for one of the pairs (r_1, θ_1) or (r_2, θ_2)).

Thus, a strategy in our sense is a set F of quadruples of the form $(t, r, \theta, C[\,])$, where r is a rule, θ is an admissible substitution for r, $C[\,]$ is an admissible context for (r, θ), such that $t = C[LHS(r)\theta]$.

Now, if R is a CERS and F a strategy for R, then the encoding R_F of the strategy F in R is defined as follows. We take the signature of R for the signature of R_F; we take the rules of R for R_F-rules without changing the sets of admissible assignments for them; finally, for each rule r and admissible assignment θ, if and only if $(t, r, \theta, C[\,]) \in F$, we declare $C[\,]$ to be admissible for (r, θ) in R_F.

Of course, our encoding is neither the only nor always the best way of encoding strategies. For example, in the case of a CERS R and the innermost strategy F, it would be more natural to define the corresponding system R_F as having the same alphabet, the same rewrite rules, and the same admissible contexts for the pairs (r, θ) such that $\theta \in AA(r)$ in R_F, but $AA_{R_F}(r)$ in R_F should be a subset of $AA_R(r)$ in R such that $\theta \in AA(r)$ belongs to $AA_{R_F}(r)$ iff the values of all metavariables of r under θ are in R-nf. Recall that strategies are often given also by transition rules; examples are the lazy and call-by-value λ-calculus. This is just another way of defining admissible contexts.

In general, for a strategy F for some orthogonal CERS R, R_F need not be orthogonal again. Examples are the innermost and needed strategies.

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 [BBKW89]. 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 are 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 be expressed as a conditional ERS. In [BBKW89] 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

A Hilbert style system H, as defined e.g. in [Sho67], has a number of axioms $F_1, F_2, \ldots$ and two rules: *Modus Ponens*, allowing to infer B when A and $A \to B$ 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 formulas

$$G_1, G_2, G_3, \ldots, G_m \tag{1}$$

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. In order to model proofs in the system H, to each proof (1) in H we associate a reduction sequence, in an appropriate CERS R_H, starting from the term P^0 and leading to the term $P^m(G_1, \ldots, G_m)$, where P^m is an m-ary operator symbol of type $Bool \times \ldots \times Bool \to Nat$. P^n encodes a list of m formulae. The alphabet of the CERS R_H consists of the alphabet of H augmented by the symbols P^n. The rules, more precisely the rule-schemata, of R_H are:

1. $P^n(A_1, \ldots, A_n) \to P^{n+1}(A_1, \ldots, A_n, F)$, for each n. 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.

2. $P^n(A_1, \ldots, A_k, \ldots, A_k \to A, \ldots, A_n) \to P^{n+1}(A_1, \ldots, A_k, \ldots, A_k \to A, \ldots, A_n, A)$ for each $n \geq 2$. The A_k may also appear after $A_k \to A$ in the sequence. Admissible substitutions are the same as in the previous case.

3. $P^n(A_1, \ldots, (A/x)A_k, \ldots, A_n) \to P^{n+1}(A_1, \ldots, (A/x)A_k, \ldots, A_n, \exists a A_k)$ for each $n \geq 1$. An admissible assignment θ assigns arbitrary formulae to $A_1, \ldots, A_n$, and a term to A.

[6] The left-linearity condition in [BBKW89] is redundant, since it is implied by essential nonambiguity.

Note the presence of the metasubstitution $(A/x)A_k$ in the LHS of the 3-rules. This kind of rule is allowed in our CERSs; the LHSs of the rules are not necessarily simple metaterms.

Thus, rewriting generates lists of theorems, the latest theorem derived always being the last formula of the list. A list can be viewed as a proof of its last theorem, but it does not determine a unique proof in the way it is encoded here. This can easily be overcome, when needed, by coding the information which rule was applied and to which theorems, into the system (this is done in the first encoding of Gentzen style proof systems into CERSs, in the next section).

4.5 Encoding of Gentzen style proof systems

Encoding a Gentzen style proof system (or a sequent calculus) 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.

A sequent calculus G has rules of one of the two following forms, where Γ and Δ are lists of formulas (we ignore structural rules, which do not cause problems for our encoding):

$$r : \frac{\Gamma' \vdash \Delta'}{\Gamma \vdash \Delta}, \qquad r : \frac{\Gamma' \vdash \Delta',\ \Gamma'' \vdash \Delta''}{\Gamma \vdash \Delta}$$

We present a sequent $F_1, \ldots, F_n \vdash G_1, \ldots, G_m$ from G as a term $P_{n,m}(F_1, \ldots, F_n, G_1, \ldots, G_m)$, where $P_{n,m}$ is a function symbol of arity $m + n$. A proof is presented as $r(sequ, pr_1)$ or $r(sequ, pr_1, pr_2)$, depending on the number of premises in the G-rule, where r is the name of the last applied rule, considered as a binary or ternary function symbol, $sequ$ is the root of the proof tree in G, and pr_i are representations of the proofs whose roots have been used as the premises in the last step of the G-inference. In a stage of a G-inference several subproofs $SPR_1, \ldots, SPR_k$ are constructed. We represent that stage as $PS^k(pr_1, \ldots, pr_k)$, where pr_i is the representation of SPR_i.

The CERS R_G that encodes G has three kinds of rewrite rule-schemata, which correspond to introduction of axioms, and to applications of one and two premise rules, respectively:

1. $PS^k(pr_1, \ldots, pr_k) \to PS^{k+1}(pr_1, \ldots, pr_k, ax(axiom, \emptyset))$;
2. $PS^k(pr_1, \ldots, r'(sequ', pr'), \ldots, pr_k) \to PS^k(pr_1, \ldots, r(sequ, r'(sequ', pr')), \ldots, pr_k)$;
3. $\begin{array}{l} PS^k(pr_1, \ldots, r'(sequ', pr'), r''(sequ'', pr''), \ldots, pr_k) \to \\ \qquad PS^k(pr_1, \ldots, r(sequ, r'(sequ', pr'), r''(sequ'', pr'')), \ldots, pr_k); \end{array}$

Here $axiom$ is the representation of an axiom of G – a sequent of the form $\Gamma \vdash \Gamma$, $sequ$ represents the sequent $\Gamma \vdash \Delta$, and similarly for $sequ'$ and $sequ''$. For example, the rewrite schema for the G-rule

$$\forall^+ : \frac{\Gamma \vdash F_1, \ldots, F[y/x], \ldots, F_n}{\Gamma \vdash F_1, \ldots, \forall x F, \ldots, F_n}$$

where y doesn't occur free in the lower sequent, has the form

$$\begin{array}{l} PS^k(pr_1, \ldots, r'(P_{n,m}(\Gamma, F_1, \ldots, F[y/x], \ldots, F_n),\ pr'), \ldots, pr_k) \to \\ \qquad PS^k(pr_1, \ldots, \forall^+(P_{n,m}(\Gamma, F_1, \ldots, \forall x F, \ldots, F_n), r'(P_{n,m}(\Gamma, F_1, \ldots, F_n), pr')), \ldots, pr_k) \end{array}$$

where m is the number of formulas in the list Γ and the same condition on y is required. Now a G-inference is encoded as an R_G-reduction starting from the constant PS^0. From each term in the reduction sequence one can recover the whole history of the corresponding G-inference. When this is not necessary, one can use a simpler encoding, where subproofs $SPR_1, \ldots, SPR_k$ in G are represented as $PS^k(sequ_1, \ldots, sequ_k)$, and $sequ_i$ is the above representation of the root sequent of SPR_i. In this encoding, we need the following rewrite rule-schemata:

1. $PS^k(sequ_1, \ldots, sequ_k) \to PS^{k+1}(sequ_1, \ldots, sequ_k, axiom)$;
2. $PS^{k+1}(sequ_1, \ldots, sequ', \ldots, sequ_k) \to PS^k(sequ_1, \ldots, sequ, \ldots, sequ_k)$;
3. $PS^k(sequ_1, \ldots, sequ', sequ'', \ldots, sequ_k) \to PS^k(sequ_1, \ldots, sequ, \ldots, sequ_k)$;

A G-inference is again encoded as a reduction starting from the constant PS^0, and one can again recover the history of the G-inference now using the whole reduction sequence.

We can similarly translate other sequent calculi such as classical or intuitionistic linear logic, natural deduction systems (since they can be given in the form of sequent calculi (see Prawitz [Pra71]), etc. Further, in order to translate say Curry's typed λ-calculus the same method can be applied – it is enough to represent a statement $F : t$ (meaning that a formula F has the type t) e.g., by $T(F, t)$; as mentioned above, expressions that require many sorted alphabets can be written in CERSs using the notion of admissible terms.

Differently from the Curry-Howard correspondence where reduction corresponds to proof-simplification (cut-elimination), we have chosen to encode the proof process, i.e. reduction corresponds to proof-search.

4.6 Encoding of the π-calculus

In this section, we will encode the version of π-calculus as described in [Mil92] as a CERS. Recall that π-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 [Mil92]).

To π-calculus a CERS (Σ_π, R_π) can be associated as follows. The alphabet Σ_π 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 π-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_π of the form

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

1. P, Q, X, Z are metavariables, and admissible assignments for X, Z are variables.

2. 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).

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

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

5 Conclusions

We have introduced context-sensitive conditional ERSs. These are a natural generalisation of unconditional ERSs what we have shown by presenting a smooth generalisation of the theory of orthogonality to the context-sensitive conditional case. Furthermore, we have shown the expressive power of the framework by encoding several rewriting systems occurring in computer science in it. An expressive framework allowing for natural encodings of interesting systems provides a means for comparing these systems. We think CERSs can have a purpose in this.

References

[Aka93] Akama Y. On Mints' reduction for ccc-calculus. In: Proc. of the 1^{st} International conference on Typed Lambda Calculus and Applications, TLCA'93, Springer LNCS, vol. 664, Bezem M., Groote J.F., eds., 1993, p. 1–12.

[AFMOW94] 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.

[BBKW89] 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.

[Bar84] Barendregt H.P. The Lambda Calculus, its Syntax and Semantics. North-Holland, 1984.

[BK86] Bergstra J. A., Klop J. W. Conditional Rewrite Rules: confluence and termination. JCSS vol. 32, n. 3, 1986, p. 323–362.

[DOS88] Dershowitz N., Okada M., Sivakumar G. Canonical conditional rewrite systems. In: Proc. of the 9^{th} International Conference on Automated Deduction, Springer LNCS, vol. 310, p. 538–549.

[HL91] Huet G., Lévy J.-J. Computations in Orthogonal Rewriting Systems. In Computational Logic, Essays in Honour of Alan Robinson, ed. by J.-L. Lassez and G. Plotkin, MIT Press, 1991.

[Kha88] Khasidashvili Z. β-reductions and β-developments of λ-terms with the least number of steps. In: Proc. of the International Conference on Computer Logic COLOG'88, Tallinn 1988, Springer LNCS, v. 417, P. Martin-Löf and G. Mints, eds. 1990, p. 105–111.

[Kha92] Khasidashvili Z. The Church-Rosser theorem in Orthogonal Combinatory Reduction Systems. Report 1825, INRIA Rocquencourt, 1992.

[Kha93] Khasidashvili Z. Optimal normalization in orthogonal term rewriting systems. In: Proc. of the 5^{th} International Conference on Rewriting Techniques and Applications, RTA'93, Springer LNCS, vol. 690, C. Kirchner, ed. Montreal, 1993, p. 243-258.

[Kha94] 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, Springer LNCS, vol. 813, Nerode A., Matiyasevich Yu. V. eds. St. Petersburg, 1994. p. 191–203.

[Klo80] Klop J.W. Combinatory Reduction Systems. Mathematical Centre Tracts 127. Centre for Mathematics and Computer Science, Amsterdam, 1980.

[Klo90] Klop J.W. Term Rewriting Systems. Report CS-R9073, Centre for Mathematics and Computer Science, 1990.

[Klo92] Klop J.W. Term Rewriting Systems. In: S.Abramsky, D.Gabby, and T.Maibaum eds. Handbook of Logic in Computer Science, vol. II, Oxford University Press, 1992.

[KOR93] 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.

[Lor93] Loría-Sáenz C.A. A Theoretical Framework for Reasoning about Program Construction Based on Extensions of Rewrite Systems. Ph. D. Thesis, Universität Kaiserslautern, 1993.

[Mel93] Melliès P.-A. An abstract theorem of finite developments. Talk presented at CONFER-meeting, september 1993, Amsterdam.

[Mes92] Meseguer J. Conditional Rewriting Logic as a unified model of concurrency. Journal of TCS 96, 1992, p. 73–155.

[Mil92] Milner R. Functions as processes. In: Journal of Mathematical structures in Computer Science. 2(2): 1992, p. 119–141.

[Ned73] Nederpelt R.P. Strong normalization for a typed lambda-calculus with lambda structured types. Ph.D. Thesis, Eindhoven, 1973.

[Nip93] 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, Springer LNCS, vol. 664, Bezem M., Groote J.F., eds., 1993, p. 306-317.

[Oos94] Van Oostrom V. Confluence for Abstract and Higher-Order Rewriting. Ph. D. Thesis, Free University of Amsterdam, 1994.

[OR94] 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, Springer LNCS, vol. 813, Nerode A., Matiyasevich Yu. V., eds. St. Petersburg, 1994, p. 379-392.

[Pra71] Prawitz D. Ideas and results in proof theory. In: Proc. of the 2^{nd} Scandinavian Logic Symposium, Fenstad L.E. ed. 1971, p. 235-307.

[vR93] Van Raamsdonk F. Confluence and superdevelopments. In: Proc. of the 5^{th} International Conference on Rewriting Techniques and Applications, RTA'93, Springer LNCS, vol. 690, C. Kirchner, ed. Montreal, 1993, p. 168-182.

[Sho67] Shoenfield J. R. Mathematical Logic. Addison Wesley, 1967.

[Tak93] Takahashi M. λ-Calculi with Conditional Rules. In: Proc. of the 1^{st} International conference on Typed Lambda Calculus and Applications, TLCA'93, Springer LNCS, vol. 664, Bezem M., Groote J.F., eds., 1993, p. 406–417.

[Toy88] Toyama Y. Confluent term rewriting systems with membership conditions. In: Proc. of the 1^{st} International Workshop on Conditional Term Rewriting Systems, Springer LNCS, vol. 308, 1988, p. 128–141.

[Wol93] Wolfram D. The clausal theory of types. Cambridge Tracts in Theoretical Computer Science, vol. 21, Cambridge University Press, 1993.