Residuals in Higher-Order Rewriting

H. J. Sander Bruggink

Department of Philosophy, Utrecht University
Email: bruggink@phil.uu.nl
Homepage: http://www.phil.uu.nl/~bruggink

Abstract. Residuals have been studied for various forms of rewriting and residual systems have been defined to capture residuals in an abstract setting. In this article we study residuals in orthogonal Pattern Rewriting Systems (PRSs). First, the rewrite relation is defined by means of a higher-order rewriting logic, and proof terms are defined that witness reductions. Then, we have the formal machinery to define a residual operator for PRSs, and we will prove that an orthogonal PRS together with the residual operator mentioned above, is a residual system. As a side-effect, all results of (abstract) residual theory are inherited by orthogonal PRSs, such as confluence, and the notion of permutation equivalence of reductions.

1 Introduction

This paper deals with residual theory: what remains of a reduction after another reduction from the same object has been performed? Let φ and ψ be reductions. Intuitively, the residual of φ after ψ, written φ/ψ, should consist of exactly those steps of φ which were not in ψ. In the literature, residuals have been studied in various degrees of abstraction [2–4, 6, 8, 13, 14], and for various forms of reduction (e.g. reduction in the λ-calculus, first-order term rewriting, and concurrency theory). In this paper we study residuals in a subclass of Higher-order Rewriting Systems (HRSs), orthogonal Pattern Rewriting Systems (orthogonal PRSs).

Even in first-order term rewriting, calculating residuals is a non-trivial task. Performing a reduction may duplicate the redexes of other reductions, thus potentially increasing the length of their residuals. In the higher-order case, the problems caused by duplication are more severe: now, copies of the same redex may get nested. Consider the orthogonal PRS which consists of the following two rules:

$$\begin{aligned} \mu &: \lambda z.\mathsf{mu}(\lambda x.z(x)) \to \lambda z.z(\mathsf{mu}(\lambda x.z(x))) \\ \rho &: \qquad \lambda x.f(x) \to \lambda x.h(x,x) \end{aligned}$$

Consider the term $s = \mathsf{mu}(\lambda x.f(x))$. The rule μ can be applied to the whole term (because $(\lambda z.\mathsf{mu}(\lambda x.z(x)))(\lambda x.f(x)) =_\beta s$) and the rule ρ can be applied to the subterm $\lambda x.f(x)$, so the following steps exists from s:

$$\begin{aligned} \varphi &: \mathsf{mu}(\lambda x.f(x)) \to f(\mathsf{mu}(\lambda x.f(x))) \\ \psi &: \mathsf{mu}(\lambda x.f(x)) \to \mathsf{mu}(\lambda x.h(x,x)) \end{aligned}$$

The residual of ψ after φ is the reduction

$$\begin{aligned} f(\mathsf{mu}(\lambda x.f(x))) &\to\ h(\mathsf{mu}(\lambda x.f(x)), \mathsf{mu}(\lambda x.f(x)) \\ &\to^*\ h(\mathsf{mu}(\lambda x.h(x,x)), \mathsf{mu}(\lambda x.h(x,x))) \end{aligned}$$

in which we see that one copy of the ρ-redex duplicates another (nested) copy of the ρ-redex.

In this paper we define a projection operator for proof terms, which are witnesses to multistep reductions. The operator projects one proof term over another and returns the residual of that proof term after the other. We define the projection operator by means of an inference system (postponing the proof that it is actually defined on orthogonal PRSs to the last part of the paper), prove that a PRS with projection operator is a residual system, and give an algorithm which calculates residuals.

An extended version of this paper was made available as technical report nr. 221 at `http://preprints.phil.uu.nl/lgps/`.

2 Preliminaries

2.1 Higher-Order Rewriting

We use Higher-order Rewriting Systems (HRSs) [7]. In fact, we consider HRSs as HORSs [12] with the simply typed λ-calculus as substitution calculus. We presuppose working knowledge of the λ-calculus, but in this section we will quickly recall the important notions of HRSs.

We fix in advance a signature Σ of simply typed constants (over a set of base types $\mathcal{B}$)[1]. *Preterms* are simply typed λ-terms over Σ. We identify α-equivalent preterms. We consider $\beta\eta$-equivalence classes of preterms. Since it is well-known that β-reduction combined with restricted η-expansion ($\beta\overline{\eta}$-reduction) is both confluent (modulo α-equivalence) and strongly normalizing, we can consider $\beta\overline{\eta}$-normal forms as unique representatives of the $\beta\eta$-equivalence classes. So, we define: *terms* are preterms in $\beta\overline{\eta}$-normal form. A *context* C is a term of the form $\lambda x.C_0$, such that x occurs free in C_0 exactly once.

We write stu for $(st)u$, and we use, for arbitrary (pre)terms $s, t_1, \ldots, t_n$, the following notation: $s(t_1, \ldots, t_n) = st_1 \ldots t_n$. Often, s will just be a function symbol, but the same notation is used if s is a term of the form $\lambda x_1 \ldots x_n.s_0$.

A term s is a *pattern* if all of its free variables x occur in some subterm of s of the form $x(y_1, \ldots, y_n)$, where the y_i are *distinct* bound variables.

Definition 2.1. *A* rewrite rule *is a tuple $l = \lambda x_1 \ldots x_n.l_0 \to \lambda x_1 \ldots x_n.r_0 = r$, where l (the left-hand side) and r (the right-hand side) are closed terms of the same type, and l is not η-equivalent to a variable. The rule is* left-linear *if $x_1, \ldots, x_n$ occur in l_0 exactly once.*

A Higher-order Rewrite System *(HRS) $\mathcal{H}$ is a set of rewrite rules. $\mathcal{H}$ is left-linear, if all its rules are. An HRS is a* Pattern Rewrite System *(PRS) if, for all of its rules $\lambda x_1 \ldots x_n.l_0 \to \lambda x_1 \ldots x_n.r_0$, l_0 is a pattern.*

[1] All definitions must be read as having the signature as an implicit parameter.

Let $\mathcal{H}$ be an HRS. We define the rewrite relation $\to_{\mathcal{H}}$ as follows [16]: s rewrites to t, written $s \to_{\mathcal{H}} t$ (the subscript is omitted if clear from the context), if there is a context C and a rule $l \to r \in R$, such that $s \twoheadleftarrow_\beta C(l)$ and $C(r) \twoheadrightarrow_\beta t$. By $\to_{\mathcal{H}}^*$ we denote the reflexive, transitive closure of $\to_{\mathcal{H}}$.

The most important reason one might have to use PRSs, is the following result of Miller [10]: unification of patterns is decidable, and if two patterns are unifiable, a most general unifier can be computed. This entails that the rewriting relation induced by a PRS is decidable.

We mention the following property of higher-order rewriting. It is non-trivial due to the implicit β-reductions in su and tv. Proofs can be found in [7, 12].

Proposition 2.2. *Let s, t, u, v be terms. If $s \to^* t$ and $u \to^* v$ then $su \to^* tv$.*

2.2 Residual Theory

Residual theory was studied in, among others, [2–4, 6, 8]. In this section, we present residuals in an abstract setting, following [13, 14], which was, in turn, based on [17]. If φ and ψ are reductions from the same object, in an arbitrary form of rewriting, then what can we tell in general of what the residual of φ after ψ must look like?

The most general form of rewriting system, which, for that reason, we will use in this section, is an *abstract rewriting system* (ARS). An ARS is a structure $\mathcal{R} = \langle A, R, \mathrm{src}, \mathrm{tgt}\rangle$ where A is a set of objects, R is a set of steps, and src and tgt are functions from R to A, specifying the source and target of the steps, respectively. Two steps are called *coinitial* if they start at the same object.

Definition 2.3. *A* residual system *is specified by a triple $\langle \mathcal{R}, 1, / \rangle$ where: $\mathcal{R}$ is an (abstract) rewriting system; 1 is a function from objects (of $\mathcal{R}$) to steps, such that $\mathrm{src}(1(s)) = \mathrm{tgt}(1(s)) = s$; and $/$, the projection function, is a function from pairs of coinitial steps to steps, with $\mathrm{src}(\varphi/\psi) = \mathrm{tgt}(\psi)$ and $\mathrm{tgt}(\varphi/\psi) = \mathrm{tgt}(\psi/\varphi)$, such that the following identities hold:*

$$\begin{gathered} 1/\varphi = 1 \\ \varphi/1 = \varphi \\ \varphi/\varphi = 1 \\ (\varphi/\psi)/(\chi/\psi) = (\varphi/\chi)/(\psi/\chi) \end{gathered}$$

The result of projecting φ over ψ (i.e. φ/ψ) is called the *residual* of φ after ψ. The intuitions behind the first three identities and the requirements to sources and targets are immediately clear. Noting that if we want to project φ over ψ and then over χ, we actually have to project φ over ψ and then over χ/ψ to make sure that the steps are coinitial, the last identity just states that projecting φ over ψ and then over χ yields the same result as projecting φ over ψ and χ in reverse order.

Theorem 2.4. *If $\langle \mathcal{R}, 1, / \rangle$ is a residual system, then $\mathcal{R}$ is confluent.*

Proof. Let $\langle \mathcal{R}, 1, / \rangle$ be a residual system, φ a step from a to b and ψ a step from a to c. Then ψ/φ is a step from b to some d and φ/ψ a step from c to the same object d.

Residual theory provides an elegant formalization of the notion of equivalence of reductions: two reductions are the same if the residual of the one after the other is an empty reduction, and vice versa. This formalization is called *permutation equivalence*. We define, for reductions φ, ψ:

$$\varphi \lesssim \psi \text{ if } \varphi/\psi = 1$$
$$\varphi \simeq \psi \text{ if } \varphi \lesssim \psi \text{ and } \psi \lesssim \varphi$$

It is not difficult to prove that $\lesssim$ is a quasi-order, and $\simeq$ is a congruence.

One of the side-effects of the main result of the paper, is that the above notion of permutation equivalence transfers directly to PRSs. Laneve & Montanari [5] give an axiomatic definition of permutation equivalence for the related format of orthogonal Combinatory Reduction Systems (CRSs), by translating CRS to first-order TRS and then using a first-order rewrite logic. We apply a higher-order rewrite logic to PRSs directly.

3 Higher-Order Rewrite Logic

In this section we give an alternative definition of the rewrite relation by means of a higher-order rewrite logic, i.e. a higher-order equational logic (see e.g. [11, 19]) without the symmetry rule (cf. [9]). The rules of the higher-order rewrite logic are presented in Table 1, together with witnessing proof terms ($\rho : l \to r$ is a rule, and a is an arbitrary function symbol or variable). Note that $l(s_1, \ldots, s_n)$ is implicitly reduced to $\beta\overline{\eta}$-normal form. The rules don't include a symmetry rule; this rule can be easily simulated by the other rules, and is therefore left out. Note that the rule and apps rules function as axioms if $n = 0$. We write $s \geq t$ if there is a proof term φ such that $\varphi : s \geq t$.

Proposition 3.1. *$s \to^* t$ iff $s \geq t$.*

Proof. The left-to-right case of the proposition is trivial, and the right-to-left case is done by structural induction on the inference of $s \geq t$.

In the rest of the paper, the following conventions are used: f, g range over function symbols, x, y range over variables, a, b range over function symbols *and* variables, and ρ, θ are rule symbols, where l, r are the left- and right hand side of ρ. Suppose $\varphi : s \geq t$. The terms s and t will be called the source and target of φ, respectively, and we introduce the functions $\mathrm{src}(\varphi) = s$ and $\mathrm{tgt}(\varphi) = t$. It is easily seen that $s : s \geq s$. Thus, we define the unit function 1 as $1(t) = t$. We will write 1 for each reduction which is the unit of some term; usually the exact term can be found by looking at the source or the target.

Proof terms are convenient, because they are terms, and so we have technical machinery to deal with them [1, 13, 14]. We relate proof terms to the conventional

$$\frac{\varphi_1 : s_1 \geq t_1 \ldots \varphi_n : s_n \geq t_n}{\rho(\varphi_1, \ldots, \varphi_n) : l(s_1, \ldots, s_n) \geq r(t_1, \ldots, t_n)}\ \mathsf{rule}$$

$$\frac{\varphi_1 : s_1 \geq t_1 \ldots \varphi_n : s_n \geq t_n}{a(\varphi_1, \ldots, \varphi_n) : a(s_1, \ldots, s_n) \geq a(t_1, \ldots, t_n)}\ \mathsf{apps}$$

$$\frac{\varphi : s \geq t}{\lambda x.\varphi : \lambda x.s \geq \lambda x.t}\ \mathsf{abs}$$

$$\frac{\varphi : s \geq u \quad \psi : u \geq t}{(\varphi \cdot \psi) : s \geq t}\ \mathsf{trans}$$

Table 1. Rewrite logic for HRSs with witnessing proof terms

rewriting terminology in the following way: a *multistep* (or just *step* for short) is a proof term which contains no $\cdot$'s; a *proper step* is a multistep with only one rule symbol in it, and a (multistep) *reduction* is a proof term of the form $\varphi_1 \cdot \ldots \cdot \varphi_n$ (modulo associativity of $\cdot$), where the φ_i are multisteps. Note that these notions intuitively correspond with the usual non proof term based notions.

We associate to each HRS $\mathcal{H}$ the following ARS $\hat{\mathcal{H}}$: terms are its objects, the *proof terms* of $\mathcal{H}$ are its *steps*, and the src and tgt functions simply are the ones introduced above. The translation of $\mathcal{H}$ into $\hat{\mathcal{H}}$ will be done implicitly.

4 Higher-Order Term Residual Systems

From now on, we restrict our attention to PRSs. Let a pre-slash-dot term be a proof term over an extended signature which includes a polymorphic *projection operator* $/ : \alpha \to \alpha \to \alpha$ (note that every proof term is a slash-dot term as well). Slash-dot terms are pre-slash-dot terms modulo the following equations:

$$\begin{aligned} f(\varphi_1 \cdot \psi_1, \ldots, \varphi_n \cdot \psi_n) &= f(\varphi_1, \ldots, \varphi_n) \cdot f(\psi_1, \ldots, \psi_n) \\ \lambda x.(\varphi \cdot \psi) &= \lambda x.\varphi \cdot \lambda x.\psi \\ 1 \cdot \varphi &= \varphi \\ \varphi \cdot 1 &= \varphi \end{aligned}$$

The first two of the equations are called the *functorial identities*, and the last two are called the *unit identities*.

We are interested in defining a projection function which associates to each slash-dot term the proof term which represents the desired residual reduction. We do this by first defining a simplification relation, and then proving that the 'normal forms' of this relation are proof terms, and unique for each slash-dot term. The projection function is then the function which associates to each slash-dot term this normal form.

4.1 A First Attempt

Simplification of terms is usually modelled as a rewriting system. In [13, 14], a rewriting system is presented which reduces (first-order) slash-dot terms to their corresponding proof term. The naive method to transfer the system to the higher-order case, is to add the following rule:

$$(\lambda x.\varphi'(x)/\lambda x.\psi'(x))z \to \varphi'(z)/\psi'(z)$$

This rule pushes abstractions outwards. The variable z is used to correctly handle bound variables. However, the rule is not equipped to handle nesting correctly. Consider the following two-rule PRS:

$$\begin{aligned} \mu : \lambda z.\mathsf{mu}(\lambda x.z(x)) &\to \lambda z.z(\mathsf{mu}(\lambda x.z(x))) \\ \rho : \qquad \lambda x.f(x) &\to \lambda x.g(x) \end{aligned}$$

and the following steps:

$$\begin{aligned} \mathsf{mu}(\lambda x.\rho(x)) : \mathsf{mu}(\lambda x.f(x)) &\geq \mathsf{mu}(\lambda x.g(x)) \\ \mu(\lambda x.f(x)) : \mathsf{mu}(\lambda x.f(x)) &\geq f(\mathsf{mu}(\lambda x.f(x))) \end{aligned}$$

Wee see that the proof term $\mathsf{mu}(\lambda x.\rho(x))/\mu(\lambda x.f(x))$ reduces in a number of steps to $\rho(\mathsf{mu}(\lambda x.\rho(x)/\lambda x.\rho(x)))$, and then in the final step the two copies of $\mathsf{mu}(\lambda x.\rho(x))$, which are not supposed to be further reduced, 'cancel each other out', resulting in the (incorrect) proof term $\rho(\mathsf{mu}(\lambda x.f(x)))$. Changing the fifth rule into

$$(\lambda x.\varphi'(x)/\lambda x.\psi'(x))z \to \varphi'(\bot z)/\psi'(\bot z)$$

where $\bot$ is a new symbol which makes sure that applications of the other rules are blocked, and adding rules to make sure that $\bot\varphi/\bot\varphi \to^* \varphi$, seems, at first sight, to solve the problem, but I have chosen another approach which I find more elegant.

4.2 Residuals of Compatible Reductions

We define the 'simplification' relation $\succcurlyeq$ between slash-dot terms and proof terms by means of the inference system Res given in Table 2 on page 7. We write $\vdash^{\mathcal{K}} \varphi \succcurlyeq \chi$ to denote that the inference $\mathcal{K}$ has $\varphi \succcurlyeq \chi$ as its final conclusion. The function $|\mathcal{K}|$ returns the 'depth' of an inference, i.e. if $|\mathcal{K}|$ is an inference with immediate subinferences $\mathcal{L}_1, \ldots, \mathcal{L}_n$, then $|\mathcal{K}| = \max_{0<i\leq n} |\mathcal{L}_i| + 1$. We write $\vdash^k \varphi \succcurlyeq \chi$ if an inference $\mathcal{K}$ exists such that $\vdash^{\mathcal{K}} \varphi \succcurlyeq \chi$ and $|\mathcal{K}| \leq k$. If k is omitted, $\mathcal{K}$ may be of arbitrary size, and in this case the $\vdash$ will often be omitted as well. The *principal rule* of an inference $\mathcal{K}$ is the last rule which is applied, i.e. the rule which appears at the bottom of the inference. We will assume the function $\mathsf{pr}(\mathcal{K})$ which returns the principal rule of an inference $\mathcal{K}$.

Residual rules:

$$\frac{\varphi_1/\psi_1 \succcurlyeq \chi_1 \cdots \varphi_n/\psi_n \succcurlyeq \chi_n}{a(\varphi_1,\ldots,\varphi_n)/a(\psi_1,\ldots,\psi_n) \succcurlyeq a(\chi_1,\ldots,\chi_n)}\, R_1$$

$$\frac{\varphi_1/\psi_1 \succcurlyeq \chi_1 \cdots \varphi_n/\psi_n \succcurlyeq \chi_n}{\rho(\varphi_1,\ldots,\varphi_n)/\rho(\psi_1,\ldots,\psi_n) \succcurlyeq r(\chi_1,\ldots,\chi_n)}\, R_2$$

$$\frac{\varphi_1/\psi_1 \succcurlyeq \chi_1 \cdots \varphi_n/\psi_n \succcurlyeq \chi_n}{\rho(\varphi_1,\ldots,\varphi_n)/l(\psi_1,\ldots,\psi_n) \succcurlyeq \rho(\chi_1,\ldots,\chi_n)}\, R_3$$

$$\frac{\varphi_1/\psi_1 \succcurlyeq \chi_1 \cdots \varphi_n/\psi_n \succcurlyeq \chi_n}{l(\varphi_1,\ldots,\varphi_n)/\rho(\psi_1,\ldots,\psi_n) \succcurlyeq r(\chi_1,\ldots,\chi_n)}\, R_4$$

$$\frac{\varphi/\psi \succcurlyeq \chi}{\lambda x.\varphi/\lambda x.\psi \succcurlyeq \lambda x.\chi}\, R_5$$

$$\frac{\varphi_1/\psi \succcurlyeq \varphi_1' \quad \psi/\varphi_1 \succcurlyeq \psi' \quad \varphi_2/\psi' \succcurlyeq \varphi_2'}{(\varphi_1\cdot\varphi_2)/\psi \succcurlyeq \varphi_1'\cdot\varphi_2'}\cdot L \qquad \frac{\varphi/\psi_1 \succcurlyeq \varphi' \quad \varphi'/\psi_2 \succcurlyeq \chi}{\varphi/(\psi_1\cdot\psi_2) \succcurlyeq \chi}\cdot R$$

$$\frac{\varphi \succcurlyeq \varphi' \quad \psi \succcurlyeq \psi' \quad \varphi'/\psi' \succcurlyeq \chi}{\varphi/\psi \succcurlyeq \chi}\,\mathsf{r{+}t/}$$

Replacement rules:

$$\frac{\varphi_1 \succcurlyeq \psi_1 \cdots \varphi_n \succcurlyeq \psi_n}{a(\varphi_1,\ldots,\varphi_n) \succcurlyeq a(\psi_1,\ldots,\psi_n)}\,\mathsf{repl}_a \qquad \frac{\varphi \succcurlyeq \psi}{\lambda x.\varphi \succcurlyeq \lambda x.\psi}\,\mathsf{repl}_\lambda$$

$$\frac{\varphi_1 \succcurlyeq \psi_1 \cdots \varphi_n \succcurlyeq \psi_n}{\rho(\varphi_1,\ldots,\varphi_n) \succcurlyeq \rho(\psi_1,\ldots,\psi_n)}\,\mathsf{repl}_\rho \qquad \frac{\varphi_1 \succcurlyeq \psi_1 \quad \varphi_2 \succcurlyeq \psi_2}{\varphi_1\cdot\varphi_2 \succcurlyeq \psi_1\cdot\psi_2}\,\mathsf{repl}_\cdot$$

Table 2. The inference rules for Res.

Example 4.1. Consider the PRS from Sect. 4.1. The current framework yields the correct result:

$$\cfrac{\cfrac{\cfrac{x/x \succcurlyeq x}{\rho(x)/f(x) \succcurlyeq \rho(x)}\, R_1}{\lambda x.\rho(x)/\lambda x.f(x) \succcurlyeq \lambda x.\rho(x)}\, R_5}{\mathsf{mu}(\lambda x.\rho(x))/\mu(\lambda x.f(x)) \succcurlyeq \rho(\mathsf{mu}(\lambda x.\rho(x)))}\, R_4$$

A slash-dot term φ is called *internally compatible* if there is a χ such that $\varphi \succcurlyeq \chi$. The source and target of an internally compatible slash-dot term φ with $\varphi \succcurlyeq \chi$ are defined as $\mathrm{src}(\varphi) = \mathrm{src}(\chi)$ and $\mathrm{tgt}(\varphi) = \mathrm{tgt}(\chi)$. Two slash-dot terms φ and ψ are *compatible* if φ/ψ is internally compatible. A PRS $\mathcal{H}$ is called compatible if all possible pairs of proof terms φ, ψ of $\mathcal{H}$ are compatible.

The following lemma expresses, in a sense, that proof terms are the 'final objects' of the relation $\succcurlyeq$ defined by the inference system.

Lemma 4.2. *Let φ be a proof term. Then: $\vdash^{\mathcal{K}} \varphi \succcurlyeq \psi$ if and only if $\varphi = \psi$.*

Proof. ($\Rightarrow$) by induction on $\mathcal{K}$, and ($\Leftarrow$) by induction on the length of φ.

Next, we prove a few standardization properties of the proposed inference system, which will come in handy in the later proofs. Given a desired outcome, Lemma 4.3 and Lemma 4.4 are used to select the principal rule of a valid inference with the desired conclusion (if it exists).

Lemma 4.3. *Suppose $\vdash^{\mathcal{K}} \varphi/\psi \succcurlyeq \chi$.*

1. *If $\varphi = a(\varphi_1, \ldots, \varphi_n)$ and $\psi = a(\varphi_1, \ldots, \varphi_n)$, then there is an inference $\mathcal{K}'$ with $|\mathcal{K}'| \leq |\mathcal{K}|$ such that $\vdash^{\mathcal{K}'} \varphi/\psi \succcurlyeq \chi$ and $\mathsf{pr}(\mathcal{K}') = R_1$.*
2. *If $\varphi = \rho(\varphi_1, \ldots, \varphi_n)$ and $\psi = \rho(\varphi_1, \ldots, \varphi_n)$, then there is an inference $\mathcal{K}'$ with $|\mathcal{K}'| \leq |\mathcal{K}|$ such that $\vdash^{\mathcal{K}'} \varphi/\psi \succcurlyeq \chi$ and $\mathsf{pr}(\mathcal{K}') = R_2$.*
3. *If $\varphi = \rho(\varphi_1, \ldots, \varphi_n)$ and $\psi = l(\varphi_1, \ldots, \varphi_n)$, then there is an inference $\mathcal{K}'$ with $|\mathcal{K}'| \leq |\mathcal{K}|$ such that $\vdash^{\mathcal{K}'} \varphi/\psi \succcurlyeq \chi$ and $\mathsf{pr}(\mathcal{K}') = R_3$.*
4. *If $\varphi = l(\varphi_1, \ldots, \varphi_n)$ and $\psi = \rho(\varphi_1, \ldots, \varphi_n)$, then there is an inference $\mathcal{K}'$ with $|\mathcal{K}'| \leq |\mathcal{K}|$ such that $\vdash^{\mathcal{K}'} \varphi/\psi \succcurlyeq \chi$ and $\mathsf{pr}(\mathcal{K}') = R_4$.*
5. *If $\varphi = \lambda x.\varphi_0$ and $\psi = \lambda x.\psi_0$, then there is an inference $\mathcal{K}'$ with $|\mathcal{K}'| \leq |\mathcal{K}|$ such that $\vdash^{\mathcal{K}'} \varphi/\psi \succcurlyeq \chi$ and $\mathsf{pr}(\mathcal{K}') = R_5$.*

Proof. By induction on $|\mathcal{K}|$.

Lemma 4.4. *Suppose $\vdash^{\mathcal{K}} \varphi/\psi \succcurlyeq \chi$.*

1. *If $\varphi = \varphi_1 \cdot \varphi_2$, then there is an inference $\mathcal{K}'$ with $|\mathcal{K}'| \leq |\mathcal{K}|$ such that $\vdash^{\mathcal{K}'} \varphi/\psi \succcurlyeq \chi$ and $\mathsf{pr}(\mathcal{K}) = \cdot L$.*
2. *If $\psi = \psi_1 \cdot \psi_2$, then there is an inference $\mathcal{K}'$ with $|\mathcal{K}'| \leq |\mathcal{K}|$ such that $\vdash^{\mathcal{K}'} \varphi/\psi \succcurlyeq \chi$ and $\mathsf{pr}(\mathcal{K}) = \cdot R$.*

Proof. By induction on $|\mathcal{K}|$.

Lemma 4.5. *If $\varphi \succcurlyeq \chi$ and $\varphi \succcurlyeq \chi'$, then $\chi = \chi'$.*

Proof. By induction on the sum of the sizes of the inferences.

We define the relation $\approx$ to be the reflexive, symmetric and transitive closure of $\succcurlyeq$. By Lemma 4.5 and the fact that if $\varphi \succcurlyeq \chi$ then χ is a proof term (easily proved by induction), we can take proof terms as the unique representatives of the classes of $\approx$-equivalent slash-dot terms. We can now define the projection operator // as follows: $\varphi \,//\, \psi = \chi$ if χ is the unique representative of the slash-dot term φ/ψ. Theorem 4.6 is proved in Sect. 4.3.

Theorem 4.6. *$\langle \mathcal{H}, 1, // \rangle$ is a residual system, if $\mathcal{H}$ is a compatible PRS.*

Corollary 4.7. *A compatible PRS is confluent.*

Proof. By Theorems 4.6 and 2.4.

4.3 Proof of Theorem 4.6

In this subsection we prove Theorem 4.6, i.e. we show that a compatible PRS together with unit and projection operator is a residual system. We mention the following two auxiliary lemmas, of which the proof is easy:

Lemma 4.8.

1. $(\varphi \cdot \psi)/\chi \approx \varphi/\chi \cdot \psi/(\chi/\varphi)$
2. $\chi/(\varphi \cdot \psi) \approx (\chi/\varphi)/\psi$

Lemma 4.9. *$\approx$ is a congruence.*

To prove that we are dealing with a residual system, we have to show that sources and targets match (Prop. 4.10), and that the residual axioms hold (Prop. 4.11).

Proposition 4.10. *Sources and targets match, i.e.:*

1. $\mathrm{src}(\varphi/\psi) = \mathrm{tgt}(\psi)$
2. $\mathrm{tgt}(\varphi/\psi) = \mathrm{tgt}(\psi/\varphi)$

Proof. By induction on the inferences of $\varphi/\psi \succcurlyeq \chi$ and $\psi/\varphi \succcurlyeq \xi$ we easily prove that $\mathrm{src}(\chi) = \mathrm{tgt}(\psi)$ and $\mathrm{tgt}(\chi) = \mathrm{tgt}(\xi)$.

Proposition 4.11. *The residual axioms hold, i.e.:*

1. $1/\varphi \approx 1$
2. $\varphi/1 \approx \varphi$
3. $\varphi/\varphi \approx 1$
4. $(\varphi/\psi)/(\chi/\psi) \approx (\varphi/\chi)/(\psi/\chi)$

Proof. (1)–(3) are proved by induction on the length of φ. In addition, (2) is based on (1), and (3) on (1) and (2).

In order to prove (4) we introduce the *layered size* $|\varphi|$ of a slash-dot term φ:

$$\begin{aligned} |f(\varphi_1, \ldots, \varphi_n)| &= 1 + \max_{0<i\leq n} |\varphi_i| \\ |x(\varphi_1, \ldots, \varphi_n)| &= 1 + \max_{0<i\leq n} |\varphi_i| \\ |\rho(\varphi_1, \ldots, \varphi_n)| &= 1 + \max_{0<i\leq n} |\varphi_i| \\ |\lambda x.\varphi| &= |\varphi| \\ |\varphi \cdot \psi| &= |\varphi| + 1 + |\psi| \\ |\varphi/\psi| &= |\varphi| \end{aligned}$$

Now (4) is verified by induction on the sum of the layered sizes of φ, ψ and χ.

The proof follows the same pattern as the one in [13]. Suppose that either φ, ψ or χ is a composite. If φ is a composite, we have the following, where the various (underlined) steps follow from Lemma 4.9 and either the induction hypothesis or Lemma 4.8:

$$
\begin{array}{ll}
& \underline{((\varphi_1 \cdot \varphi_2)/\psi)}/(\chi/\psi) \\
\approx & \underline{((\varphi_1/\psi) \cdot (\varphi_2/(\psi/\varphi_1)))/(\chi/\psi)} \\
\approx & \underline{(\varphi_1/\psi)/(\chi/\psi)} \cdot ((\varphi_2/(\psi/\varphi_1))/\underline{((\chi/\psi)/(\varphi_1/\psi))}) \\
\approx_{\mathrm{IH}} & \underline{(\varphi_1/\chi)/(\psi/\chi)} \cdot \underline{((\varphi_2/(\psi/\varphi_1))/((\chi/\varphi_1)/(\psi/\varphi_1)))} \\
\approx_{\mathrm{IH}} & \underline{(\varphi_1/\chi)/(\psi/\chi) \cdot ((\varphi_2/(\chi/\varphi_1))/((\psi/\varphi_1)/(\chi/\varphi_1)))} \\
\approx & \underline{((\varphi_1/\chi) \cdot (\varphi_2/(\chi/\varphi_1)))}/(\psi/\chi) \\
\approx & ((\varphi_1 \cdot \varphi_2)/\chi)/(\psi/\chi)
\end{array}
$$

If ψ is a composite, we do:

$$
\begin{array}{ll}
& \underline{(\varphi/(\psi_1 \cdot \psi_2))}/\underline{(\chi/(\psi_1 \cdot \psi_2))} \\
\approx & \underline{((\varphi/\psi_1)/\psi_2)}/\underline{((\chi/\psi_1)/\psi_2)} \\
\approx_{\mathrm{IH}} & \underline{((\varphi/\psi_1)/(\chi/\psi_1))}/(\psi_2/(\chi/\psi_1)) \\
\approx_{\mathrm{IH}} & \underline{((\varphi/\chi)/(\psi_1/\chi))/(\psi_2/(\chi/\psi_1))} \\
\approx & (\varphi/\chi)/\underline{(\psi_1/\chi \cdot \psi_2/(\chi/\psi_1))} \\
\approx & (\varphi/\chi)/((\psi_1 \cdot \psi_2)/\chi)
\end{array}
$$

The case that χ is a composite, is the inverse of this.

Now consider the case that none of φ, ψ, χ are composites. Suppose that $\varphi = f(\boldsymbol{\varphi})$, $\psi = f(\boldsymbol{\psi})$, and $\chi = f(\boldsymbol{\chi})$, where we use the notation $\boldsymbol{x}$ for the vector $x_1, \ldots, x_n$. By Lemma 4.3, the following inference must exist:

$$
\dfrac{\dfrac{\cdots \quad \overline{\varphi_i/\psi_i \succcurlyeq \zeta_{1,i}} \quad \cdots}{f(\boldsymbol{\varphi})/f(\boldsymbol{\psi}) \succcurlyeq f(\boldsymbol{\zeta_1})} \quad \dfrac{\cdots \quad \overline{\chi_i/\psi_i \succcurlyeq \zeta_{2,i}} \quad \cdots}{f(\boldsymbol{\chi})/f(\boldsymbol{\psi}) \succcurlyeq f(\boldsymbol{\zeta_2})} \quad \dfrac{\cdots \quad \overline{\zeta_{1,i}/\zeta_{2,i} \succcurlyeq \xi_{1,i}} \quad \cdots}{f(\boldsymbol{\zeta_1})/f(\boldsymbol{\zeta_2}) \succcurlyeq f(\boldsymbol{\xi_1})}}{(f(\boldsymbol{\varphi})/f(\boldsymbol{\psi}))/(f(\boldsymbol{\chi})/f(\boldsymbol{\psi})) \succcurlyeq f(\boldsymbol{\xi_1})}
$$

and similarly we obtain an inference of $(f(\boldsymbol{\varphi})/f(\boldsymbol{\chi}))/(f(\boldsymbol{\psi})/f(\boldsymbol{\chi})) \succcurlyeq f(\boldsymbol{\xi_2})$. Using the same subinferences for $\varphi_i/\psi_i \succcurlyeq \zeta_{1,i}$, $\chi_i/\psi_i \succcurlyeq \zeta_{2,i}$ and $\zeta_{1,i}/\zeta_{2,i} \succcurlyeq \xi_{1,i}$, we easily obtain $(\varphi_i/\psi_i)/(\chi_i/\psi_i) \succcurlyeq \xi_{1,i}$, and similarly we show $(\varphi_i/\chi_i)/(\psi_i/\chi_i) \succcurlyeq \xi_{2,i}$. Since, by induction hypothesis, $(\varphi_i/\psi_i)/(\chi_i/\psi_i) \approx (\varphi_i/\chi_i)/(\psi_i/\chi_i)$, we know now that $\xi_{1,i} \approx \xi_{2,i}$, so there are $\xi_{3,i}$ such that $\xi_{1,i} \succcurlyeq \xi_{3,i} \preccurlyeq \xi_{2,i}$. Two easy inferences prove $f(\boldsymbol{\xi_1}) \succcurlyeq f(\boldsymbol{\xi_3})$ and $f(\boldsymbol{\xi_2}) \succcurlyeq f(\boldsymbol{\xi_3})$. We put everything together with transitivity and get:

$$
(f(\boldsymbol{\varphi})/f(\boldsymbol{\psi}))/(f(\boldsymbol{\chi})/f(\boldsymbol{\psi})) \approx f(\boldsymbol{\xi_3}) \approx (f(\boldsymbol{\varphi})/f(\boldsymbol{\chi}))/(f(\boldsymbol{\psi})/f(\boldsymbol{\chi}))
$$

The same strategy works in the other non-composite cases, e.g. if $\varphi = \rho(\boldsymbol{\varphi})$, $\psi = \rho(\boldsymbol{\psi})$, and $\chi = r(\boldsymbol{\chi})$, since the difficult nesting problems (duplicating behaviour within right-hand sides of rules) occur only on the right of the $\succcurlyeq$ symbols.

Now, Theorem 4.6 follows from Prop. 4.10 and Prop. 4.11. QED.

4.4 Computing Residuals

In Sect. 4.2 only a specification of the simplification relation was given, but Lemma 4.3 and Lemma 4.4 already hinted at the existence of an algorithm which

effectively computes the representative of a slash-dot term. In this section, such an algorithm is indeed given. We also prove that it terminates for two special cases of slash-dot term, and show that if the algorithm terminates, it prints the correct answer.

Definition 4.12. *The (recursive) function* $\mathrm{sim}(\pi)$ *on proof terms* π*, is defined by the following pseudo-program:*

$\mathrm{sim}((\varphi_1/\varphi_2)/\psi)) = \mathrm{sim}(\varphi'/\psi)$
  where $\varphi' = \mathrm{sim}(\varphi_1/\varphi_2)$
$\mathrm{sim}(\varphi/(\psi_1/\psi_2)) = \mathrm{sim}(\varphi/\psi')$
  where $\psi' = \mathrm{sim}(\psi_1/\psi_2)$
$\mathrm{sim}(x(\varphi_1,\ldots,\varphi_n)/x(\psi_1,\ldots,\psi_n)) = x(\mathrm{sim}(\varphi_1/\psi_1),\ldots,\mathrm{sim}(\varphi_n/\psi_n))$
$\mathrm{sim}(f(\varphi_1,\ldots,\varphi_n)/f(\psi_1,\ldots,\psi_n)) = f(\mathrm{sim}(\varphi_1/\psi_1),\ldots,\mathrm{sim}(\varphi_n/\psi_n))$
$\mathrm{sim}(\rho(\varphi_1,\ldots,\varphi_n)/\rho(\psi_1,\ldots,\psi_n)) = r(\mathrm{sim}(\varphi_1/\psi_1),\ldots,\mathrm{sim}(\varphi_n/\psi_n))$
$\mathrm{sim}(\rho(\varphi_1,\ldots,\varphi_n)/l(\psi_1,\ldots,\psi_n)) = \rho(\mathrm{sim}(\varphi_1/\psi_1),\ldots,\mathrm{sim}(\varphi_n/\psi_n))$
$\mathrm{sim}(l(\varphi_1,\ldots,\varphi_n)/\rho(\psi_1,\ldots,\psi_n)) = r(\mathrm{sim}(\varphi_1/\psi_1),\ldots,\mathrm{sim}(\varphi_n/\psi_n))$
$\mathrm{sim}(\lambda x.\varphi/\lambda x.\psi) = \lambda x.(\varphi/\psi)$
$\mathrm{sim}((\varphi_1\cdot\varphi_2)/\psi)) = \varphi_1'\cdot\varphi_2'$
  where $\varphi_1' = \mathrm{sim}(\varphi_1/\psi)$
    $\varphi_2' = \mathrm{sim}(\varphi_2/\psi')$
    where $\psi' = \mathrm{sim}(\psi/\varphi_1)$
$\mathrm{sim}(\varphi/(\psi_1\cdot\psi_2)) = \mathrm{sim}(\varphi'/\psi_2)$
  where $\varphi' = \mathrm{sim}(\varphi/\psi_1)$
$\mathrm{sim}(f(\varphi_1,\ldots,\varphi_n)) = f(\mathrm{sim}(\varphi_1),\ldots,\mathrm{sim}(\varphi_n))$
$\mathrm{sim}(\rho(\varphi_1,\ldots,\varphi_n)) = \rho(\mathrm{sim}(\varphi_1),\ldots,\mathrm{sim}(\varphi_n))$
$\mathrm{sim}(\lambda x.\varphi) = \lambda x.\mathrm{sim}(\varphi)$
if *none of the above cases apply* **then**
  print *"incompatible"*

Proposition 4.13.

1. *If* φ *and* ψ *are* reductions*, then* $\mathrm{sim}(\varphi/\psi)$ *terminates.*
2. *If* φ *is* internally compatible*, then* $\mathrm{sim}(\varphi)$ *terminates.*

Proof. We prove the first item first. If φ and ψ are reductions, then the computation of $\mathrm{sim}(\varphi/\psi)$ proceeds in two stages: first the compositions on the outside of the terms are dealt with, and in this stage the number of composition symbols in the proof term strictly decreases in each step; and then, when φ and ψ are parallel steps, the length of the proof term strictly decreases in each step.

Secondly, if φ is internally compatible, then an inference $\mathcal{K}$ exists such that $\vdash^{\mathcal{K}} \varphi/ \succcurlyeq \chi$. The second item can be proved by induction on $\mathcal{K}$, using Lemma 4.3 and Lemma 4.4.

Termination *in general* is hard to show. If a proof term φ is not internally compatible, an inference of $\varphi \succcurlyeq \chi$ is not at our disposal, so we cannot use induction on the inference. The problem is then the cases which deal with composition. In these cases the size of the terms which are passed recursively to the function, may actually be larger than the size of the term under consideration.

Conjecture 4.14. $\mathrm{sim}(\varphi)$ terminates for *all* slash-dot terms φ.

The main result of the paper does not depend on this conjecture, although, because of its not being proved, a small detour has to be followed in Sect. 5.1.

Proposition 4.15. $\mathrm{sim}(\varphi) = \chi$ *if and only if* $\varphi \succcurlyeq \chi$.

Proof. The 'only if' side is proved by recursively building an inference of $\varphi \succcurlyeq \chi$. The 'if' side is easily proved by using Lemma 4.3 and Lemma 4.4.

5 Orthogonality

In this section we relate compatibility with the well-known notion of orthogonality. In order to define orthogonality, we need to define overlap, and this is done by associating with each proper step a set of redex positions, and then looking at the intersection of the redex positions of two coinitial proper steps.

Positions are sequences of natural numbers. If P is a set of positions, and p is a position, we write $p \star P$ for $\{pq \mid q \in P\}$. First, we need to define the set of all positions of a term. Let $\Box$ denote the empty context.

$$\begin{aligned}
\mathcal{P}os(\Box) &= \emptyset \\
\mathcal{P}os(x(s_1, \ldots, s_n)) &= \{\epsilon\} \cup \textstyle\bigcup_{0<i\leq n} i \star \mathcal{P}os(s_i) \\
\mathcal{P}os(f(s_1, \ldots, s_n)) &= \{\epsilon\} \cup \textstyle\bigcup_{0<i\leq n} i \star \mathcal{P}os(s_i) \\
\mathcal{P}os(\lambda x.s) &= \{\epsilon\} \cup 1 \star \mathcal{P}os(s)
\end{aligned}$$

where x is a variable and f a function symbol.

Now, let φ be a proper step. We define the set of redex positions of φ, written $\mathcal{RP}os(\varphi)$, as:

$$\begin{aligned}
\mathcal{RP}os(x(\varphi_1, \ldots, \varphi_n)) &= \textstyle\bigcup_{0<i\leq n} i \star \mathcal{RP}os(\varphi_i) \\
\mathcal{RP}os(f(\varphi_1, \ldots, \varphi_n)) &= \textstyle\bigcup_{0<i\leq n} i \star \mathcal{RP}os(\varphi_i) \\
\mathcal{RP}os(\lambda x.\varphi_0) &= 1 \star \mathcal{RP}os(\varphi_0) \\
\mathcal{RP}os(\rho(\varphi_1, \ldots, \varphi_n)) &= \mathcal{P}os(l(\Box, \ldots, \Box))
\end{aligned}$$

Note that, since φ is a proper step, in the last equation there are no more rule symbols in the φ_i.

Two coinitial proper steps φ and ψ are said to be *overlapping* if $\mathcal{RP}os(\varphi) \cap \mathcal{RP}os(\psi) \neq \emptyset$. A left-linear PRS is *orthogonal*, if all pairs of different, coinitial proper steps are non-overlapping.

This definition has an infinite flavour: there are infinitely many steps one has to check. Fortunately, it is well-known that an equivalent notion of orthogonality exists, based on critical pairs [7]. Since a finite PRS has only finitely many possible critical pairs, this makes the question whether a PRS is orthogonal or not decidable. We stick to the step-based definition for convenience.

5.1 Compatibility Is Orthogonality

In this subsection we will prove that compatibility and orthogonality coincide. The difficult part of the proof, but also the most important one, is to show that orthogonality implies compatibility. One way of doing so is by contraposition: we run the algorithm of Def. 4.12 and analyse in which situations it prints *incompatible*, and show that the PRS is not orthogonal in each of these cases. There is one problem: we have not succeeded in proving that the algorithm actually always terminates, so we have to follow a small detour: we transform the incompatible proof terms into incompatible reductions, and then feed those to the algorithm.

Lemma 5.1. *If $l(\varphi_1, \ldots, \varphi_n) \cdot \rho(\psi_1, \ldots, \psi_n)$ is compatible with χ, then $\rho(\varphi_1 \cdot \psi_1, \ldots, \varphi_n \cdot \psi_n)$ is compatible with χ.*

Proof. By constructing an inference.

Theorem 5.2. *Let $\mathcal{H}$ be an PRS. $\mathcal{H}$ is orthogonal, if and only if $\mathcal{H}$ is compatible.*

Proof. We first prove the right-to-left implication. Assume that all coinitial reductions are compatible. This implies that all coinitial multisteps φ, ψ are compatible, i.e. an inference $\mathcal{K}$ exists such that $\vdash^{\mathcal{K}} \varphi/\psi \succcurlyeq \chi$. We easily prove, by induction on $\mathcal{K}$, that φ, ψ are non-overlapping.

To show the left-to-right implication, assume, by contraposition, that φ, ψ are coinitial, but not compatible. Consider the (meta-level) rewrite system which consists of all rules of the form

$$\rho(\varphi_1 \cdot \psi_1, \ldots, \varphi_n \cdot \psi_n) \Rightarrow l(\varphi_1, \ldots, \varphi_n) \cdot \rho(\psi_1, \ldots, \psi_2)$$

where $\rho : l \to r$ is a rule. It is not difficult to see that this rewrite system is strongly normalizing, and that its normal forms are actually reductions. So, applying this rewriting system to φ and ψ yields reductions φ', ψ', respectively. By Prop. 4.13, $\mathrm{sim}(\varphi'/\psi')$ terminates, and by (the contraposition of) Lemma 5.1, φ' and ψ' are not compatible.

Let φ_0/ψ_0 be the slash-dot term which was passed to sim in the last step before it terminated; φ_0 and ψ_0 must be multisteps. By Prop. 4.15 the algorithm prints *incompatible*. By coinitiality of φ_0 and ψ_0, it cannot cannot be the case that $\varphi_0 = f(\varphi_1, \ldots, \varphi_n)$ and $\psi_0 = g(\psi_1, \ldots, \psi_n)$, where $f \neq g$. So, the following must apply: $\varphi_0 = \rho(\varphi_1, \ldots, \varphi_n)$ and $\psi_0 \neq_{\beta\eta} l(\psi_1, \ldots, \psi_n)$. There are two possible causes of this. The first is that ψ_0 has a rule symbol within the redex pattern of l. But then overlapping, coinitial proper steps φ_0' and ψ_0' can be constructed by replacing all rule symbols, except the overlapping ones, of φ_0 and ψ_0, respectively, by their left-hand sides. The second possible cause is that one of the ψ_i occurs twice in $l(\psi_1, \ldots, \psi_n)$. However, then l cannot be left-linear. Both cases imply non-orthogonality. (The third 'cause' is that ψ_0 has a $\cdot$ inside the redex pattern of φ_0, but this cannot happen because compositions are moved outwards over function symbols and abstractions by the functorial identities, and l consists only of function symbols and abstractions.) The same argument can be applied to the symmetrical case.

5.2 Residuals of Orthogonal PRSs

In this subsection we prove the main result of the paper, namely that an orthogonal PRS, together with the unit and projection operator, forms a residual system. The hard work has already been done; we just need to put together the results obtained so far.

Theorem 5.3. *If $\mathcal{H}$ is an orthogonal PRS, then $\langle \mathcal{H}, 1, /\!/ \rangle$ is a residual system.*

Proof. By Theorem 5.2, $\mathcal{H}$ is compatible, and thus, by Theorem 4.6, $\langle \mathcal{H}, 1, /\!/ \rangle$ is a residual system.

It is well-known that orthogonal PRSs are confluent, as was proved in, among others, [7, 12, 15]. Here, we obtain a new proof based on the residual theory developed in this paper. The proof emerges as a simple corollary of the main result.

Corollary 5.4. *Orthogonal PRSs are confluent.*

Proof. Let $\mathcal{H}$ be an orthogonal PRS. By Theorem 5.3, $\mathcal{H}$ is a residual system, and thus by Theorem 2.4, $\mathcal{H}$ is confluent.

6 Concluding Remarks

In this paper, we have shown that orthogonal PRSs form a residual system. As a consequence, all results for residual systems are inherited, such as the notion of permutation equivalence and confluence. We have also given an algorithm which simplifies slash-dot terms to proof terms, and we have proven, in two special cases, that the algorithm terminates.

For the future, the following research is interesting. Firstly, it is interesting to find a proof (or a refutation) of the claim that the algorithm mentioned in the previous paragraph does *always* terminate. Not only is this interesting in its own right, it is my view that such a proof may aid us in the understanding of termination of higher-order rewriting, and provide new proof methods.

Secondly, it is interesting to see if the framework can be generalized to non-orthogonal, left-linear PRSs, or even arbitrary PRSs. For this to work, an error symbol must be added, to indicate non-compatibility. For the first-order case, the same approach was succesfully applied to left-linear TRSs in [13].

Acknowledgements

I wish to thank Vincent van Oostrom and the anonymous referees for their valuable remarks on preliminary versions of this paper.

References

1. Barnaby P. Hilken. Towards a proof theory of rewriting: The simply typed 2λ-calculus. *Theoretical Computer Science*, 170:407–444, 1996.
2. Gérard Huet. Residual theory in λ-calculus: A formal development. *Journal of Functional Programming*, 4(3):371–394, 1994.
3. Gérard Huet and Jean-Jacques Lévy. Computations in orthogonal rewriting systems, part I + II. In J.L. Lassez and G.D. Plotkin, editors, *Computational Logic – Essays in Honor of Alan Robinson*. MIT Press, 1991.
4. Zurab Khasidashvili and John Glauert. Relating conflict-free stable transition systems and event models via redex families. *Theoretical Computer Science*, 286(1):65–95, 2002.
5. Cosimo Laneve and Ugo Montanari. Axiomatizing permutation equivalence. *Mathematical Structures in Computer Science*, 6(3):219–215, 1996.
6. Jean-Jacques Lévy. *Réductions correctes et optimales dans le λ-calcul*. Thèse de doctorat d'état, Université Paris VII, 1978.
7. Richard Mayr and Tobias Nipkow. Higher-order rewrite systems and their confluence. *Theoretical Computer Science*, 192:3–29, 1998.
8. Paul-André Melliès. Axiomatic rewriting theory VI: Residual theory revisited. In Sophie Tison, editor, *13th International Conference on Rewriting Techniques and Applications*, pages 24–50, 2002.
9. José Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96:73–155, 1992.
10. Dale Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. *Journal of Logic and Computation*, 1(4), 1991.
11. Tobias Nipkow and Christian Prehofer. Higher-order rewriting and equational reasoning. In W. Bibel and P. Schmitt, editors, *Automated Deduction — A Basis for Applications, Volume I: Foundations*, number 8 in Applied Logic Series, pages 399–430. Kluwer Academic Press, 1998.
12. Vincent van Oostrom. *Confluence for Abstract and Higher-Order Rewriting*. PhD thesis, Vrije Universiteit, Amsterdam, 1994.
13. Vincent van Oostrom and Roel de Vrijer. *Equivalence of Reductions*, chapter 8 of [18]. 2003.
14. Vincent van Oostrom and Roel de Vrijer. Four equivalent equivalences of reductions. In *Proceedings of WRS'02 (ENTCS 70.6)*, 2003. Downloadable at: `http://www.elsevier.nl/locate/entcs/`.
15. Femke van Raamsdonk. *Confluence and Normalisation for Higher-Order Rewriting*. PhD thesis, Vrije Universiteit, Amsterdam, 1996.
16. Femke van Raamsdonk. Higher-order rewriting. In *10th International Conference on Rewriting Techniques and Applications*, 1999.
17. Eugene W. Stark. Concurrent transition systems. *Theoretical Computer Science*, 64(3):221–269, 1989.
18. Terese. *Term Rewriting Systems*. Number 55 in Camb. Tracts in Theor. Comp. Sc. Cambridge University Press, 2003.
19. D. A. Wolfram. *The Clausal Theory of Types*. Number 21 in Camb. Tracts in Theor. Comp. Sc. Cambridge University Press, 1993.