ELSEVIER

Theoretical Computer Science 175 (1997) 159–181

Theoretical Computer Science

Developing developments

Vincent van Oostrom *

Department of Mathematics and Computer Science, Vrije Universiteit, De Boelelaan 1081a, 1081 HV Amsterdam, Netherlands

Abstract

In the absence of termination, confluence of rewriting systems is often hard to establish. The class of orthogonal rewriting systems is the main class of not-necessarily-terminating, but confluent rewriting systems. The reason why confluence holds in orthogonal rewriting systems is the absence of the so-called critical pairs, making that rewrite steps never interfere (in a destructive way) with one another. We discuss some ways to adapt the *confluence by orthogonality* proof method to rewriting systems having 'innocent' critical pairs. Confluence results are obtained for lambda calculus with *beta*, *eta* and *omega* rules, lambda calculi with *restricted expansion* rules and for (first- and higher-order) term rewriting systems for which all critical pairs are *development closed*.

1. Introduction

Suppose one starts computing the value of some expression. If at some computation stage there is a choice how to compute further, but it is possible to extend every chosen pair of computations from that point, to computations to a point of confluence, the computation system is called *confluent*. If computations always terminate, checking confluence is decidable, by checking all the so-called critical pairs between the computation rules for confluence (cf. [26, 40]). However, if computations may not terminate, confluence is undecidable.

The main positive confluence results for not-necessarily-terminating rewriting systems were established for systems in which rewrite steps never interfere with one another, e.g. Curry's combinatory logic [13] and Church's λI-calculus [10]. Note however, that although steps do not interfere, they might *discard* or *copy* (*cancel* or *duplicate* in [14]) each other, and the latter fact makes that an ordinary structural induction on terms fails for proving confluence. In the case of λI-calculus, the key notion needed for a proof by induction was found by Church and Rosser to be a *sequence of contractions on*

* E-mail: oostrom@cs.vu.nl. Most of the results in the paper were obtained during my stay at NTT Basic Research Laboratory, Information Processing Principles Research Group, 3-1 Morinosato Wakamiya, Atsugi-Shi, Kanagawa-Ken, 243-01, Japan, and reported in [43].

PII 0304-3975(96)00173-9

the parts [11, p. 475] of a term, nowadays known as a *development* (cf. [6, Definition 11.2.11] or [22, Axiom 0]). Since then, the proof method based on developments has been studied extensively, leading to many Church–Rosser (confluence) theorems for what are nowadays called *orthogonal* rewriting systems. Among those there are rewriting systems having a more abstract syntax (e.g. [25, 34]), but also systems having a more elaborate syntax (e.g. [49, 2, 32, 30, 39, 42, 48]).

In this paper we are interested in rewriting systems where the confluence by orthogonality method *does not* apply directly, i.e. where an additional twist is needed to get confluence. Some results in this area are confluence of $\lambda\beta\eta$-calculus [14, p. 137], of *linear strongly closed* and of *parallel closed* first-order term rewriting systems [26, 54], and of weakly orthogonal higher-order rewriting systems [45]. We present confluence proofs for three further (classes of) rewriting systems, viz.

(i) Lambda calculus with beta, eta and Omega rules [6, Section 15.2],

(ii) Lambda calculi with *restricted expansion rules*, for example, [1]

$$\mathtt{app}(\mathtt{abs}(x.X(x)), Y) \to_{\mathsf{beta}} X(Y)$$
$$X \to_{\overline{\mathsf{eta}}} \mathtt{abs}(x.\mathtt{app}(X,x))$$

where the expansion rule $\overline{\mathsf{eta}}$ is subject to the *restriction* that no beta-redexes may be created by it.

(iii) Left-linear higher-order term rewriting systems where all *critical pairs* (cf. [26, 40]) are *development closed*, i.e. for every critical pair (s,t), t can be obtained from s by a development $s \to\!\!\!\!\circ\, t$.

The first reproves the confluence result in [6, Ch. 15.2], the second reproves several confluence results in the literature for lambda calculi with expansion rules [3, 12, 17, 16, 18], and the third generalises the confluence results for *parallel-closed* first-order term rewriting systems in [26, 54] and extends them to the higher-order term rewriting case at the same time. We would like to stress however, that our main interest is in proving these results by adaptations of the confluence by orthogonality method.

The results we present can be conveniently expressed in any of the higher-order term rewriting formats available, e.g. either in combinatory reduction systems (CRSs [32, 33]), in expression reduction systems (ERSs [29]), or in higher-order term rewriting systems (HRSs/PRSs [40, 37]), or in the HORS-format ([42, 48]). The most important reason why we have chosen to stick to PRSs is that the notions we will rely on like matching, unification, critical pairs and orthogonality have been thoroughly studied for them in literature [35, 40, 39, 42, 37, 46, 41, 48]. [2]

We start with recapitulating the rewriting essentials needed in this paper. The next three sections (Sections 3–5) are then devoted to the three adaptations mentioned above.

[1] These are just the usual beta reduction ($(\lambda x.M)N \to M[x := N]$) and restricted eta expansion ($M \to \lambda x.Mx$) rules (cf. [3]) using higher-order notation. The side-condition 'x does not occur in X' on eta is internalised in this formulation.

[2] This is another way of saying that the paper is not self-contained.

2. Rewriting

For standard rewriting no(ta)tions we refer to [15, 31], for the syntax and semantics of the lambda calculus to [6].

Definition 1 (*ARS*). An *abstract rewriting system* (*ARS*) $\rightarrow$ is a binary relation on some set $(a, b \in) A$.

On top of common binary (infix) relation notation, we assume some special rewriting notations. The denotation of the 'repeated' notation $\twoheadrightarrow$ is the transitive-reflexive closure of the denotation of $\rightarrow$, i.e. $\twoheadrightarrow = \rightarrow^*$. Similarly, the denotation of the 'inverse' notation $\leftarrow$ is the inverse of the denotation of $\rightarrow$, i.e. $\leftarrow = \rightarrow^{-1}$, and the denotation of the 'union with the inverse' notation $\leftrightarrow$ is the symmetric closure of the denotation of $\rightarrow$, i.e. $\leftrightarrow = \leftarrow \cup \rightarrow$. If $a \rightarrow b$, then we say that a *($\rightarrow$-)rewrites to* b, and the structure $\langle a, \rightarrow, b\rangle$ is called a *$\rightarrow$-step from a to b*. A *$\rightarrow$-span* [51, Section 6.3] is a pair $(\langle a, \rightarrow, b\rangle, \langle a, \rightarrow, c\rangle)$, which we will usually write as $b \leftarrow a \rightarrow c$. A *rewrite sequence* is a (finite or infinite) sequence of rewrite steps, such that for successive steps the object to which the former rewrites is the same as the object from which the latter rewrites.

Definition 2. We recapitulate some standard properties ARSs can have.

(i) An ARS $\rightarrow$ has the *triangle* property, if for each a, there exists a step $a \rightarrow a^*$, such that if $a \rightarrow b$, then $b \rightarrow a^*$.

(ii) An ARS $\rightarrow$ has the *diamond* property, if $\leftarrow ; \rightarrow \subseteq \rightarrow ; \leftarrow$.

(iii) An ARS $\rightarrow$ is *strongly confluent*, if $\leftarrow ; \rightarrow \subseteq \rightarrow ; \twoheadleftarrow$, where ; denotes relation composition.

(iv) An ARS $\rightarrow$ is *confluent*, if $\twoheadrightarrow$ has the diamond property.

(v) An ARS $\rightarrow$ is *semi-confluent*, if $\leftarrow ; \twoheadrightarrow \subseteq \twoheadrightarrow ; \twoheadleftarrow$.

(vi) An ARS $\rightarrow$ is *locally confluent*, if $\leftarrow ; \rightarrow \subseteq \twoheadrightarrow ; \twoheadleftarrow$.

(vii) An ARS $\rightarrow$ *commutes with* an ARS $\rightsquigarrow$, if $\twoheadleftarrow ; \rightsquigarrow^* \subseteq \rightsquigarrow^* ; \twoheadleftarrow$.

An object is a *($\rightarrow$)-normal form* if no rewrite steps are possible from it. An ARS is *terminating* if no infinite rewrite sequences are possible and it is *complete* if it is confluent as well.

Except for the triangle property which was abstracted from [52, Lemma 3.2; 9, Definition 3.2.1.1], and semi-confluence due to Nipkow, the properties stem from [26].

Lemma 3. *triangle $\Longrightarrow$ diamond $\Longrightarrow$ strong confluence $\Longrightarrow$ confluence $\Longleftrightarrow$ semi-confluence $\Longrightarrow$ local confluence. Furthermore, none of the not-shown-implications between the properties holds.*

Note that the properties in the lemma are all preserved under taking the reflexive closure. The difficulty with proving confluence is the presence of transitive(-reflexive)

closures in its hypothesis $\twoheadleftarrow;\twoheadrightarrow$. Since semi-confluence implies confluence (this is the so-called Strip Lemma [6, p. 279]), one of these closures can be removed. Ridding ourselves also from the other transitive closure is much harder, even impossible in general since confluence is not equivalent to local confluence, as witnessed by the abstract rewriting system $c \leftarrow b \leftrightarrow a \rightarrow d$ [24, p. 25] (obtained by condensating an infinite example in [50], due to Rosser). However, it is possible if we find some property P of rewrite sequences such that

(i) $P(a \rightarrow b)$, for all steps $a \rightarrow b$,

(ii) if $P(a \twoheadrightarrow b)$ and $b \twoheadleftarrow a \rightarrow c$, then there exists d such that $b \twoheadrightarrow d \twoheadleftarrow c$ and $P(c \twoheadrightarrow d)$.

We then say that $\rightarrow$ is *P-commuting*. By an easy induction we get the following result.

Lemma 4. *Let $\rightarrow$ be an ARS, then $\rightarrow$ is confluent $\Longleftrightarrow$ $\rightarrow$ is P-commuting for some P.*

Although the lemma shows that we have just a reformulation of confluence, we show that taking for P the property '*is a development*' suffices for proving confluence of orthogonal rewriting systems. That choice will be the basis for our adaptations of the confluence by orthogonality method.

2.1. Higher-order rewriting systems

Rewriting systems are classified according to the structure the objects have. In higher-order rewriting systems the structures are λ-terms, i.e. containing functional abstraction. Since *any binding operation can in principle be defined in terms of functional abstraction and an ordinary operation* [14, p. 85], this can be (imprecisely) reformulated as: higher-order rewriting systems are rewriting systems containing bound variables. Typical objects such systems can manipulate are $\int_0^3 x^2 \mathrm{d}x$, $\forall x.\phi(x)$, and $\lambda x.x$.

In the literature one can find many distinct notions of higher-order rewriting [32, 29, 40, 28, 55, 52, 33, 4, 19] varying as to which λ-calculus and notation for it are employed and which restrictions are imposed on the form of the rewrite rules. A comparison of those systems and a unified approach to them one can find in [44, 42, 45, 48] and we refer the reader to these for motivation and details. Since here we are interested in presenting our proof methods, rather than in describing the general framework, we employ the concrete class of higher-order pattern rewriting systems (HRSs/PRSs) introduced in [40, 37]. PRSs are rewriting systems which employ simply typed λ-calculus with $\beta\eta\alpha$-conversion, and where left-hand sides of rewrite rules are restricted to the so-called *patterns*. We present PRSs in the style of [41, 48].

Definition 5 (*PRS*). A *pattern rewriting systems* (PRS) is a pair $(\mathscr{A}, \mathscr{R})$ consisting of an alphabet $\mathscr{A}$ and a set $\mathscr{R}$ of rewrite rules.

(i) We first define the objects of a PRS. *Simple types* are defined by the grammar: $\sigma ::= b \mid \sigma \rightarrow \sigma$, for some set ($b \in$) B of *base* types, e.g. booleans or natural numbers. A *rewrite alphabet* is a set denoted by ($\mathtt{F}, \mathtt{G} \in$) $\mathscr{A}$, the elements of which are called

function symbols such that to each function symbol F a unique simple type σ_F is associated. *Preterms* are objects s such that $s:\sigma$ for some simple type σ, can be inferred from

(sym) $F:\sigma_F$, for $F \in \mathscr{A}$,

(var) $x^\sigma:\sigma$ for termvariables x,

(app) $s:\sigma \to \tau$, $t:\sigma \Longrightarrow s(t):\tau$.

(abs) $s:\tau \Longrightarrow x^\sigma.s$[3] $:\sigma \to \tau$.

We use x, X, y, Y, ... to range over term variables. Functional notation $F(s_1,\ldots,s_m)$ is employed instead of applicative notation $F(s_1)\ldots(s_m)$ in case $m > 1$ (cf. [40]). Higher-order *terms* are obtained from the preterms by quotienting by the theory consisting of $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\eta}$ (that is, the theory $\boldsymbol{\lambda\eta}$ in [6]). To make terms concrete we use their $\beta\overline{\eta}$-normal forms as representatives (unique up to α-conversion) of the equivalence classes. Here, the rewrite relations β and $\overline{\eta}$ are generated by the rules

$$(x.s)(t) \to_\beta s[x := t]$$

$$s \to_{\overline{\eta}} x.s(x)$$

where $\overline{\eta}$ is not allowed to create β-redexes and x does not occur in s. The representative of a preterm s is denoted by $s{\downarrow}$. Notions for terms are obtained from notions for preterms by omitting the prefix 'pre' from the latter.

(ii) A *rewrite rule* is a pair $\vec{X}.l \to \vec{X}.r$ of closed terms (i.e. $\beta\overline{\eta}$-normal forms) such that

(a) $\vec{X}$ is a sequence of term variables,

(b) l and r are terms of the same base type,

(c) the first symbol or *head* of l is a function symbol, i.e. l is of the form $F(s_1,\ldots,s_m)$, for some function symbol F,

(d) l is a *pattern* [35], i.e. every occurrence of an x_k among $\vec{x}$ in l has only (representatives of) pairwise distinct variables not among $\vec{x}$ as arguments. The pattern is *linear* when each variable among $\vec{x}$ occurs exactly once in l.

In general, we will call a term l satisfying those conditions a pattern and we use $l \to r$ as a shorthand notation for the rule.

A rewrite step in a PRS transforms one term to another one, by replacing some substructure by another one according to one of the rewrite rules. It is said to *act* on that substructure. Intuitively, when two rewrite steps act on disjoint substructures (such steps are also called *well-separated*, *non-critical*, *consistent*, *compatible*, *non-ambiguous* and *non-overlapping* in literature) these steps can be performed simultaneously, giving rise to so-called *development* steps.

Definition 6 (*PRS step*). Let $\mathscr{H}$ be a PRS. For preterms C, $s_1,\ldots,s_m$, we denote by $C[\![s_1,\ldots,s_m]\!]$ the result of substituting $s_1,\ldots,s_m$ for the term variables $\Box_1,\ldots,\Box_m$ in

[3] We omit the usual λ in abstractions.

$C(\Box_1$ will be abbreviated to $\Box$). If $s =^{\text{def}} C[l_1,\ldots,l_m]$ and $t =^{\text{def}} C[r_1,\ldots,r_m]$ for rewrite rules $\aleph_1 =^{\text{def}} l_1 \to r_1, \ldots, \aleph_m =^{\text{def}} l_m \to r_m$, then there is a *development prestep* $s \multimap_{C[\aleph_1,\ldots,\aleph_m]} t$ and t is said to be obtained from s by *contracting* the *development redex* $C[\aleph_1,\ldots,\aleph_m]$. The development prestep induces the *development step* $s\downarrow \multimap_{C[\aleph_1,\ldots,\aleph_m]} t\downarrow$ between terms. The ARS $\to_{\mathscr{H}}$ associated to the PRS $\mathscr{H}$ is the relation on terms obtained by requiring m to be one, and $\Box$ to occur exactly once in C in a development prestep. Notions with this restriction are obtained from the corresponding ones without it, by omitting the word 'development'. We use u, v, w to denote redexes, and identify them with their induced steps whenever convenient.

Because of the restriction to patterns, the rewrite relation is decidable [35, 40, 46]. All first-order term rewriting systems (TRSs [15, 31]), lambda calculi [6, 7], as well as combinatory reduction systems (CRSs [32, 33]), interaction systems (ISs [4, 5]), and expression reduction systems (ERSs [29]) are naturally embeddable into PRSs (see [48]).

Example 7. Consider a TRS having one rewrite rule $2\times x \to x+x$. This yields a PRS having alphabet $2{:}\,o$, $\times,+{:}\,o\to o\to o$ and a rewrite rule $\aleph : X.\times(2,X)\to X.+(X,X)$, in shorthand notation $\times(2,X)\to +(X,X)$.

(i) An example of a development prestep for this rule is $[X.\times(2,X)](3) \multimap_{[\aleph](3)} [X.+(X,X)](3)$ Since $\Box(3)$ is in $\beta\bar{\eta}$-normal form and $\Box$ occurs once in it, the development prestep induces the step $\times(2,3)\to +(3,3)$ as it should since this is a rewrite step in the TRS as well.

(ii) Another example of a development prestep is $(y.+(y,y))([X.\times(2,X)](3)) \multimap_{(y.+(y,y))([\aleph](3))} (y.+(y,y))([X.+(X,X)](3))$. It induces the development step $s \multimap t$ between terms $s =^{\text{def}} +(\times(2,3),\times(2,3)) \equiv (y.+(y,y))([X.\times(2,X)](3))\downarrow$ and $t =^{\text{def}} +((+)(3,3),(+)(3,3)) \equiv (y.+(y,y))([X.+(X,X)](3))\downarrow$. Note that it does not yield a step because $(y.+(y,y))(\Box(3))$ can be β-reduced to $+(\Box^{o\to o}(3),\Box^{o\to o}(3))$, and although this latter term is in $\beta\bar{\eta}$-normal form $\Box$ occurs twice in it so again does not yield a step.

(iii) The development step $s \multimap t$ gives rise to two distinct *serialisations*:

(a) $s \to +(+(3,3),\times(2,3)) \to t$,

(b) $s \to +(\times(2,3),+(3,3)) \to t$.

Note that if $s \multimap_{C[\aleph_1,\ldots,\aleph_m]} t$ is a development step, $s \multimap_{C\downarrow[\aleph_1,\ldots,\aleph_m]} t$ is one as well, so there is no harm in imposing the restriction that C be a term.

On the one hand, it is not obvious that no rewrite power is lost by restricting attention to steps instead of development steps, but as exemplified, every development step can be *serialised*.

Lemma 8 (Serialisation). $\to_{\mathscr{H}} \subseteq \multimap_{\mathscr{H}} \subseteq \twoheadrightarrow_{\mathscr{H}}$.

Proof. The first inclusion holds by definition. The second one was proven for our particular definition of development in [42, Proposition 3.1.22, 3.2.11], but is well-known from literature (cf. [6, Lemma 3.2.7; 47, Proposition 8; 39, Lemma 4.3; 52, Lemma 2.2; 48, Theorem 5.2.6]). □

On the other hand, it is in general not true that any number of steps from some term can be *parallelised*, i.e. performed in one development step. In particular, this does not hold true when two steps act on the same substructure, e.g. the two possible steps from F arising from the rules $\mathtt{F} \to \mathtt{G}$ and $\mathtt{F} \to \mathtt{H}$. It is said there is a *critical pair* between the two rules. However, assuming that steps are parallelisable, confluence can be proven.

Definition 9 (*Simultaneous*). We say that l *is a pattern* (*at* C) *in* s, if l is a pattern, C and s are terms, and s is a representative of $C[\![l]\!]$. A set $l_1,\ldots,l_m$ of patterns (at $C_1,\ldots,C_m$) in s is said to be *simultaneous* (*at* C), if C is a term such that C_k is a representative of $C[\![l_1,\ldots,l_{k-1},\Box,l_{k+1},\ldots,l_m]\!]$, for each $1 \leqslant k \leqslant m$. We use $\mathscr{U}$, $\mathscr{V}$, $\mathscr{W}$ to range over both sets of simultaneous redexes and their induced development steps.

Observe that simultaneity is closed under taking subsets.

Theorem 10 (Triangle). *Let* $\mathscr{H}$ *be a left-linear PRS.*

(i) *Let* $s \multimap\!\!\to_{\mathscr{U}} s^*$ *for some set of simultaneous redexes in* s. *If* $s \multimap\!\!\to_{\mathscr{V}} t$ *for* $\mathscr{V} \subseteq \mathscr{U}$, *then* $t \multimap\!\!\to s^*$.

(ii) *If every set of redexes is simultaneous, then* $\multimap\!\!\to_{\mathscr{H}}$ *has the triangle property and* $\mathscr{H}$ *is confluent.*

Proof. (i) This is implied by the Prism Theorem (introduced in [27] for the case of lambda calculus with the beta-rule) by simply forgetting about the extra dimension of residuals. The Prism Theorem itself is implied by the Finite Developments Theorem which was shown to hold for PRSs in [45, Lemma 3.8]. (For alternative proofs of the latter theorem for higher-order rewriting formats close to PRSs, see [32, 30, 34].)

(ii) Define for a term s, s^* as the term obtained by contracting the (development redex associated to the) set of all redexes in s. This set is simultaneous by assumption and since it contains any other set of redexes in s, $\multimap\!\!\to_{\mathscr{H}}$ has the triangle property by the first item. Confluence of $\mathscr{H}$ is a direct consequence of Lemmata 3 and 8 as usual. □

The class of *orthogonal* PRSs as introduced in [39, Definition 3.10], extending definitions for TRSs (cf. [15, 31]) and for CRSs (cf. [32, 47]), derives its importance from the fact that sets of redexes are always parallelisable for rewriting systems in this class.

Definition 11 (*Orthogonal*). A PRS $\mathscr{H}$ is *orthogonal*, if

(i) it is left-linear, that is, all left-hand sides of rules are linear patterns, and

(ii) it has no critical pairs, which are defined as follows.

(a) A *way of interference* between two rewrite rules $l \to r$ and $g \to d$ is a span $t \leftarrow_{C[\boxed{l \to r}]} s \leftarrow_{D[\boxed{g \to d}]} r$ of PRS steps, such that for some pattern p (a substructure both steps act on) at C' in l and at D' in g, it holds that C' at C and D' at D are patterns in the same term.

(b) A span $t' \leftarrow_{C'[\boxed{l \to r}]} s' \leftarrow_{D'[\boxed{g \to d}]} r'$ is *more general* than $t \leftarrow_{C[\boxed{l \to r}]} s \leftarrow_{D[\boxed{g \to d}]} r$ if s' is at E in s, and C' in C and D' in D are both at E.

(c) A span $t \leftarrow_{C[\boxed{l \to r}]} s \leftarrow_{D[\boxed{g \to d}]} r$ (and also its associated pair (t,r)) is called *critical*, if it is a most general way of interference between two rules. In order to make our definition coincide with the ones found in rewriting literature, the perfect symmetry in this notion has to be removed and to that end we require $\Box$ to be the head of D. In case $\Box$ is the head of C as well, we speak of an *overlay* (interference at the root).

A *weakly* orthogonal PRS is a left-linear PRS such that all critical pairs are trivial, i.e. left- and right components of critical pairs are identical.

Our definition of critical pair looks rather different from the ones in [40, 37, Definition 4.1]. The present equivalent definition was adapted from [42, Definition 3.2.32], borrowing from [51]. Because of the restriction to patterns, critical pairs can be computed [35, 40, 46]. The natural embeddings in [48] from TRSs, CRSs, ISs and ERSs into PRSs all preserve orthogonality and we have the following result [42, Proposition 3.1.49, 3.2.25].

Theorem 12 (Simultaneity). *In an orthogonal PRS, sets of redexes are simultaneous.*

3. Beta–Eta–Omega

In this section we present a first adaptation of the confluence by orthogonality method, to the rewriting system $\mathscr{BH}\mathscr{O}$ (see [6, Section 15.2]). The function symbols of $\mathscr{BH}\mathscr{O}$ are $\mathtt{app}: o \to (o \to o), \mathtt{abs}: (o \to o) \to o$ and $\Omega: o$. *Lambda Omega* terms are terms over this alphabet defined by the grammar:

$$s ::= \Omega \mid x \mid \mathtt{app}(s,s) \mid \mathtt{abs}(x.s)$$

such that all variables are of basetype. Rewrite rules are the usual beta and eta, and Omega which collapses all unsolvable lambda terms to a unique representative Ω. (A lambda term s is *unsolvable* [6, Definition 8.3.1] if the equation $\mathtt{app}(s,\vec{X}) \leftrightarrow^*_{\mathsf{beta,eta}} \mathtt{I}$ is unsolvable in $\vec{X}$, where $\mathtt{app}(s,\vec{X})$ and $\mathtt{I}$ are abbreviations for $\mathtt{app}(\ldots\mathtt{app}(s,X_1)\ldots,X_m)$ and $\mathtt{abs}(X.X)$.)

$$\mathsf{app}(\mathsf{abs}(x.X(x)),Y) \to_{\mathsf{beta}} X(Y)$$
$$\mathsf{abs}(x.\mathsf{app}(X,x)) \to_{\mathsf{eta}} X$$
$$X \to_{\mathsf{Omega}} \Omega \text{ , if } X \text{ is unsolvable and } X \not\equiv \Omega$$

Remark 13. This is not completely accurate since the left-hand side of the Omega-rule is not a pattern, so the system is not a PRS. This is only a technical complication, because one can introduce a 'marker' symbol $U: o \to o$ into the alphabet and then the fact that some term s is unsolvable can be reflected syntactically by writing it as $U(s)$. For this to work it is essential that the condition on the rewrite rule (unsolvability) is closed under rewriting (cf. [52]), which it is [6, Lemma 15.2.4].

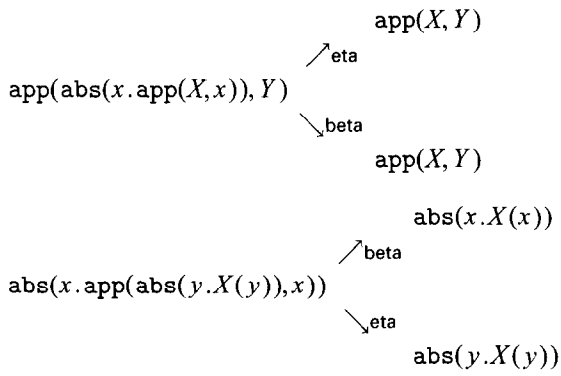
Although $\mathscr{BHO}$ is clearly not orthogonal, it does actually satisfy the triangle property, i.e. one can identify a (non-unique) maximal set of simultaneous redexes in every term. We do not show this, but the construction below is based on it.

Theorem 14 (Barendregt [6, Theorem 15.2.15(ii)]). *$\mathscr{BHO}$ is confluent.*

Proof. First remark that the condition $x \not\equiv \Omega$ in the Omega rule does not matter for confluence so can be safely omitted. We show that $\to$ is P-commuting for $P =^{\text{def}}$ *is a development*, so consider a set $\mathscr{U}$ of simultaneous redexes in s and any step $s \to_v t$. We distinguish cases according to the relative positions of v and $\mathscr{U}$. First remark that we may assume that no step in $\mathscr{U}$ is inside some Omega step in $\mathscr{U}$, since these steps will we erased anyway.

(i) If $\mathscr{U} \cup \{v\}$ is simultaneous, then the first part of Theorem 10 can be applied to $s \multimap_{\mathscr{U}\cup\{v\}} s^*$.

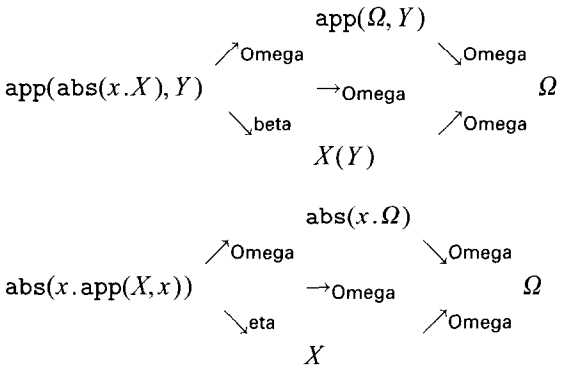
(ii) If v is a beta or eta step and interferes with some eta or beta step $u \in \mathscr{U}$, then we observe that the critical pairs



between these rules are trivial, hence $s \to_u t$ must hold as well, and switching attention to $\mathscr{U}$ and u, we can proceed as in the previous case.

(iii) If there is an overlap between v and some step u in $\mathscr{U}$, where one is a beta or eta and the other an Omega step, it must arise from one of the following critical

pairs, where we have indicated that these critical pairs are confluent. (Note that both $\mathtt{app}(\Omega, Y)$ and $\mathtt{abs}(x.\Omega)$ are unsolvable.)



Call the corresponding horizontal Omega step w. Consider $\mathscr{U} \cup \{w\}$. The only way in which this set can be not simultaneous, is if some eta or beta step in $\mathscr{U}$ interferes with the beta or eta step among $\{u,v\}$. It cannot be u by simultaneity of $\mathscr{U}$, but it cannot be v either, since then we would have been in the previous case. We conclude that $\mathscr{U} \cup \{w\}$ is simultaneous, and by the remark made, we end up in the first case. □

From confluence of this calculus, it follows that identifying all unsolvable (undefined) terms in lambda beta eta calculus preserves consistency.

4. Expansion

In the last few years, in several papers the point of view has been taken that the extensional equations in typed lambda calculi should be taken as *expansion* rules rather than as *reduction* rules. For example, instead of the eta reduction rule

$$\mathtt{abs}(x.\mathtt{app}(X,x)) \to_{\mathsf{eta}} X$$

one prefers to have its inverse, the eta expansion rule

$$X \to_{\mathsf{eta}^{-1}} \mathtt{abs}(x.\mathtt{app}(X,x))$$

The reason to prefer expansions over reductions is simple: for deciding equality in a theory, one tries to implement equality by a complete rewrite systems, and implementing the eta rule as a reduction rule was found to lead (when combined with other rules) to non-confluent systems. On the other hand, expansion rules like eta expansion obviously give rise to infinite rewrite sequences. The solution to regain termination consists in imposing the *restriction*

no beta-redexes may be created by eta-expansion

on the eta^{-1}-rule, the result of which is denoted by $\overline{\mathsf{eta}}$. As noted in [12], a somewhat remarkable fact is that just disallowing the creation of **beta** entails termination in the simply typed case. Here we will *not* be concerned with matters of termination, and restrict attention to proving confluence for typed lambda calculi with expansion rules. We present the proof method by the example of the calculus from [3]. However, our reasoning pertains to many lambda calculi with expansion rules (e.g. [36, 3, 12, 17, 16, 18]). The alphabet of the calculus consists of app, abs as before, $\pi: o \to o \to o$, $\pi_1: o \to o$, $\pi_2: o \to o$ and $*: o$. Rewriting will be on a subset of the set of all terms over the alphabet. Types (not to be confused with the simple types employed for the syntax of PRSs!) are generated by the grammar

$$A ::= T \mid A \supset A \mid A \times A$$

Typed terms are terms which are typable by the following inference system.

(i) $*: T$,
(ii) $x: A,\ s: B \implies \mathsf{abs}(x.s): A \supset B$,
(iii) $s: A \supset B,\ t: A \implies \mathsf{app}(s,t): B$,
(iv) $s: A,\ t: B \implies \pi(s,t): A \times B$,
(v) $s: A \times B \implies \pi_1(s): A,\ \pi_2(s): B$.

The rewrite rules of the calculus are

$$\begin{aligned}
\mathsf{app}(\mathsf{abs}(x.X(x)), Y) &\to_{\mathsf{beta}} X(Y)\\
X &\to_{\mathsf{eta}^{-1}} \mathsf{abs}(x.\mathsf{app}(X,x))\\
\pi_1(\pi(X,Y)) &\to_{\mathsf{pi}_1} X\\
\pi_2(\pi(X,Y)) &\to_{\mathsf{pi}_2} Y\\
X &\to_{\mathsf{pi}^{-1}} \pi(\pi_1(X), \pi_2(X))\\
X &\to_{\mathsf{tau}^{-1}} *
\end{aligned}$$

Remark 15. As in the previous section, not all the left-hand sides of the rules are patterns. In particular, left-hand sides of expansion rules are not. Whether an expansion rule can be applied to a term is only determined by its type, in other words, by the type of the edge leading to the term in its tree representation. Again this is only a technical complication, because one can introduce for every type A, a unary symbol $A: o \to o$ in the syntax, and write $A(s)$ to represent the fact that s is a term of type A. This works since types are preserved under rewriting (the subject reduction property).

The rules are classified into a set of reduction rules $\mathscr{R} =^{\mathrm{def}} \{\mathsf{beta}, \mathsf{pi}_1, \mathsf{pi}_2\}$ and a set of expansion rules $\mathscr{E} =^{\mathrm{def}} \{\mathsf{eta}^{-1}, \mathsf{pi}^{-1}, \mathsf{tau}^{-1}\}$. Reduction steps can be thought of as eliminating a so-called destructor–constructor pair (in the logical meaning, i.e. corresponding to introduction and elimination for the type) in a term. Schematically, $-d-c- \to_{\mathscr{R}} -$. Terms constructed from abs (i.e. having abs as head symbol) can be destructed by infinitely many app-terms. Terms constructed from pi can be destructed by either a pi_1 or a pi_2. $*$ cannot be destructed. Similarly, expansion rules can be

thought of as inserting a so-called constructor–destructor(s) pair at some edge in the tree representation of a term. Schematically, $- \to_{\mathscr{E}} -c-d-$ (the tau^{-1} rule creates no destructor and the pi^{-1} rule creates two destructors). Trivial rewrite cycles would therefore arise from expansion in the following situations:

(i) If the subterm at the insertion point was already in constructor format. Schematically, $-c \to_{\mathscr{E}} -c$, or $-c- \to_{\mathscr{E}} -c-d-c- \to_{\mathscr{R}} -c-$.

(ii) If in the original term, a destructor was applied to the insertion point. Schematically, $-d- \to_{\mathscr{E}} -d-c-d- \to_{\mathscr{R}} -d-$.

These situations can be avoided by imposing on the rules in $\mathscr{E}$ the condition

> A rule in $\mathscr{E}$ may only be applied if it does not create an $\mathscr{R}$-redex and it is not the identity.

giving rise to the set of *restricted* expansion rules $\overline{\mathscr{E}} =^{\text{def}} \{\overline{\mathsf{eta}}, \overline{\mathsf{pi}}, \overline{\mathsf{tau}}\}$. Now we can state the confluence result.

Theorem 16. *$\mathscr{R} \cup \overline{\mathscr{E}}$ is confluent.*

Proof. Confluence of the system $\mathscr{R} \cup \overline{\mathscr{E}}$ can be reduced to confluence of $\mathscr{R}$ and $\overline{\mathscr{E}}$ separately by the Lemma of Hindley–Rosen [31], once it is known that $\mathscr{R}$ commutes with $\overline{\mathscr{E}}$. Confluence of $\mathscr{R}$ and $\overline{\mathscr{E}}$ separately are well-known on the one hand and easily checked on the other (orthogonality applies in both cases), so we do not elaborate on it but concentrate on commutation instead. The problem with showing commutation of $\mathscr{R}$ and $\overline{\mathscr{E}}$ is that restricted expansion rules are, well, restricted (context-sensitive) and the usual ways to prove commutation are geared to unrestricted (context-free) rules. This problem was tackled in different ways by different authors, which we will comment on shortly. We solve it, by showing $\mathscr{R}$ to be $\overline{P}$-commuting for $\overline{P}(d) =^{\text{def}} d$ *is a development of a set of $\overline{\mathscr{E}}$-redexes* (in the obvious 'commutation'-version of this notion). In order to prove commutation of restricted expansion and reduction, one needs to study how they interact. First note that the unrestricted expansion rules together with the reduction rules form an orthogonal PRS, so $\mathscr{R}$ is found to be P-commuting for $P(d) =^{\text{def}} d$ *is a development of a set of $\mathscr{E}$-redexes*, i.e.

$$t \leftarrow\!\!\circ\!\!-_{\mathscr{E}} s \to_{\mathscr{R}} r \subseteq t \to\!\!\circ\!\!\to_{\mathscr{R}} p' \leftarrow\!\!\circ\!\!-_{\mathscr{E}} r$$

Since $\overline{\mathscr{E}}$-redexes are special cases of $\mathscr{E}$-redexes we have

$$t \leftarrow\!\!\circ\!\!-_{\overline{\mathscr{E}}} s \to_{\mathscr{R}} r \subseteq t \to\!\!\circ\!\!\to_{\mathscr{R}} p' \leftarrow\!\!\circ\!\!-_{\mathscr{E}} r$$

The only nasty point about the interaction between restricted expansion and reduction steps is the fact that the latter can destroy the former, that is, the development of expansion steps from r to p' may contain some expansion steps which are not restricted. The reason is that a (residual[4] of a) restricted expansion step might not be allowed

[4] The notion of residual is the standard one since [11].

anymore after an $\mathscr{R}$-step due to its context-sensitivity (see Example 18). However, the unrestricted commutation can easily be transformed into the restricted commutation

$$t \leftarrow\!\!\circ\!\!\!-_{\overline{\mathscr{E}}} s \rightarrow_{\mathscr{R}} r \subseteq t \rightarrow\!\!\circ\!\!\!-_{\mathscr{R}} p' \twoheadrightarrow_{\mathscr{R}} p \leftarrow\!\!\circ\!\!\!-_{\mathscr{E}} r$$

To see this, observe that the unrestricted expansion from r to p' can be partitioned into a restricted expansion of maximal length to some term p followed by an unrestricted expansion to p': $r \twoheadrightarrow_{\overline{\mathscr{E}}} p \twoheadrightarrow_{\mathscr{E}} p'$. We argue that this sequence can actually be chosen such that $r \rightarrow\!\!\circ\!\!\!-_{\overline{\mathscr{E}}} p \twoheadleftarrow_{\mathscr{R}} p'$. First, the $\mathscr{R}$-step from s to r may cause residuals of distinct expansions in s to be the 'on the same edge' in r, e.g. if $x, X: o \to o$, then the residuals of the steps expanding x and X, after the step $\mathsf{app}(\mathsf{abs}(x.x), X) \to_{\mathsf{beta}} X$ are both expansion steps of X. Schematically these residuals are situated as: $-x-X-$. To construct the restricted development from r we can select to develop just one of them, since its expansion will surely violate the restriction of the others. Schematically: expanding x yields $-c-d-X-$, where X cannot be expanded because of the destructor, and expanding X yields $-x-c-d-$, where x cannot be expanded because of the constructor. Note that the redexes selected in this way are (and stay) simultaneous during their development. Second, note that residuals of expansion steps violating the restriction, still violate the restriction so all the steps from p to p' do so. Because of the violation, for each step there must be zero or one $\mathscr{R}$-steps in the reverse direction. □

Remark 17. The proof is purely syntactical, i.e. it also works for non-typable terms.

Example 18. In the case of $\overline{\mathsf{eta}}$ and beta, violation of the restriction for the residual of an $\overline{\mathsf{eta}}$ step after a beta-step can happen in three ways, corresponding to the three ways in which redexes can be created by beta-contraction.

(i) $\mathsf{app}(\mathsf{abs}(x.x), \mathsf{abs}(y.y)) \to_{\overline{\mathsf{eta}}} \mathsf{app}(\mathsf{abs}(x.\mathsf{abs}(z.\mathsf{app}(x,z))), \mathsf{abs}(y.y))$ expanding x, is destroyed by $\mathsf{app}(\mathsf{abs}(x.x), \mathsf{abs}(y.y)) \to_{\mathsf{beta}} \mathsf{abs}(y.y)$.

(ii) $\mathsf{app}(\mathsf{abs}(x.x), \mathsf{abs}(y.y)) \to_{\overline{\mathsf{eta}}} \mathsf{abs}(z.\mathsf{app}(\mathsf{app}(\mathsf{abs}(x.x), \mathsf{abs}(y.y)), z))$ expanding the initial term, is destroyed by $\mathsf{app}(\mathsf{abs}(x.x), \mathsf{abs}(y.y)) \to_{\mathsf{beta}} \mathsf{abs}(y.y)$.

(iii) $\mathsf{app}(\mathsf{abs}(x.\mathsf{app}(x,Y)), X) \to_{\mathsf{eta}} \mathsf{app}(\mathsf{abs}(x.\mathsf{app}(x,Y)), \mathsf{abs}(y.\mathsf{app}(X,y)))$ expanding X, is destroyed by $\mathsf{app}(\mathsf{abs}(x.\mathsf{app}(x,Y)), X) \to_{\mathsf{beta}} \mathsf{app}(X,Y)$.

4.1. Related work on expansions

There is a lot of active research going on in the field of lambda calculi with expansion rules, especially regarding termination for expansional systems beyond system F [21]. For this reason we will only briefly indicate the relationship of our method to others.

Our proof of confluence is very close to the (earlier) one by Čubrić presented in [12]. The (technical) difference is that there *minimal* developments are employed to show commutation [12, Lemma 4.7], whereas we employ developments. It is mainly a matter of taste which one prefers.

Di Cosmo and Kesner employ a completely different technique to show commutation of restricted expansions and reduction in [17]. They show that $\mathscr{R}$ is P-commuting for $P(d) =^{\text{def}}$ *d is a restricted expansion to $\overline{\mathscr{E}}$-normal form.* In order to show this, they provide a definition of the expansion normal form of an arbitrary term, by induction on the structure of a term. The necessary properties are then shown by induction.

Akama proves completeness of Mints' system using the following lemma [3, Lemma 4]:

Lemma 19. *If $\overline{\mathscr{E}}$ and $\mathscr{R}$ are both complete rewriting systems and*

$$a \to_{\mathscr{R}} b \implies a{\downarrow_{\overline{\mathscr{E}}}} \to^{+}_{\mathscr{R}} b{\downarrow_{\overline{\mathscr{E}}}}$$

then $\overline{\mathscr{E}} \cup \mathscr{R}$ is complete ($a{\downarrow_{\overline{\mathscr{E}}}}$ denotes the $\overline{\mathscr{E}}$-normal form of a).

This lemma and Akama's completeness proof are closer to the method of Di Cosmo and Kesner, sketched in the previous paragraph, than to a proof by (minimal) developments. We prefer the latter as it gives more refined results.

Di Cosmo and Piperno [18] have introduced a method to prove completeness of lambda calculi with expansion rules based on the following commutation lemma (cf. [20, p. 47]).

Lemma 20. *Let $\to_1$ and $\to_2$ be ARSs. If $\to_1$ is terminating and*

$$\leftarrow_2; \to_1 \subseteq \to_1^+; \twoheadleftarrow_2$$

holds, then $\to_1$ and $\to_2$ commute.

After establishing termination, the lemma is used to reduce confluence to local confluence. Since this lemma is based on termination, and our results are based on orthogonality, the methods are incomparable in general. However, the confluence results in the papers mentioned can be obtained directly by applying the method presented here.

For modularity questions, like whether the combination of an expansional lambda calculus with some (left-linear) confluent TRS is confluent again, we can sometimes apply the result from [45, Theorem 3.13], stating that the union of two left-linear confluent PRSs whose rules are mutually orthogonal, is confluent. E.g. the left-linear rule for unbounded recursion

$$\mathsf{fix} \to_{\mathsf{fix}} \mathsf{abs}(\mathsf{x}.\mathsf{app}(\mathsf{x},\mathsf{app}(\mathsf{fix},\mathsf{x})))$$

where fix is a fresh constant (for every type A), is confluent on its own (it is an orthogonal PRS) so its combination with a left-linear confluent TRS and with lambda calculi (without expansion rules) is confluent by that result (cf. [16, Theorem 4.8]). In order to check that confluence holds also with the expansional rules, the following lemma can be put to use.

Lemma 21 (Request lemma, Rosen [49]). *Let $\to =^{\text{def}} \to_1 \cup \to_2$. If $\to_1$ and $\to_2$ are confluent and $\twoheadleftarrow_1 ; \twoheadrightarrow_2 \subseteq \leftrightarrow_2^*; \twoheadleftarrow_1$ then $\to$ is confluent.*

Taking $\to_1 =^{\text{def}} \to_{\mathsf{fix}}$ and $\to_2$ the rewrite relation of the other (left-linear) rules, the diagram of the lemma is proved like commutation before: fix commutes with the rewriting system with unrestricted expansions (they are orthogonal to each other); if expansions are restricted then some of the expansions have to be replaced by reductions in the reverse direction. This proves that the combination of a confluent left-linear term rewriting system, with a lambda calculus with both reduction and expansion rules, and with unbounded recursion, is confluent [16, Theorem 4.11].

We hope that the method presented in this section is flexible enough to be of general usage for proving confluence of lambda calculi with expansional rules.

5. Development closed

In this section we present the third adaptation of the confluence by orthogonality method. We show that left-linear *development closed* PRSs, i.e. PRSs such that $s \multimap t$, for every critical pair (s,t) (see the right-hand side of Fig. 1), are confluent. The precursor of this result is [26, Lemma 3.3], stating that left-linear *parallel closed* TRSs, i.e. PRSs such that $s \parallel\!\!\to t$, for every critical pair (s,t) (see the left-hand side of Fig. 1), are confluent, where $s \parallel\!\!\to t$ expresses that s rewrites to t by contraction of redexes at disjoint positions.

The proof of [26, Lemma 3.3] works by showing that $\parallel\!\!\to$ has the diamond property for parallel closed systems. Since for orthogonal TRSs $\parallel\!\!\to$ does not have the triangle property,[5] and for orthogonal PRSs even the diamond property fails, $\parallel\!\!\to$ has to be abandoned to have any hope on generalising the result to higher-order rewriting.

5.1. Huet

Lemma 22. *Let $\mathscr{H}$ be a left-linear development closed PRS; then $\multimap$ has the diamond property.*

Proof. The structure of the proof is the same as Huet's and we refer to [41] for a detailed technical account of it.[6] Here we will be content with presenting the underlying idea. First, how difficult it is to prove the diamond property for a given development span $t \leftarrow\!\!\!\multimap_{\mathscr{V}} s \multimap_{\mathscr{U}} r$ can be measured by the 'amount of interference' between the sets $\mathscr{U}$ and $\mathscr{V}$. This is just the number of function symbols in s which steps from both sets act on. (Note that since $\mathscr{U}$ ($\mathscr{V}$) is simultaneous, every function symbol in s can be acted upon by at most one step from $\mathscr{U}$ ($\mathscr{V}$).)

(i) If this amount is nought, $\mathscr{U} \cup \mathscr{V}$ is a set of simultaneous redexes and the diamond property is a trivial consequence of Theorem 10.

[5] For that reason we prefer $\multimap$ to $\parallel\!\!\to$.

[6] The proof needs the notion of *descendant* which is easy to understand on an intuitive level, but hard to formalise (cf. [27]).

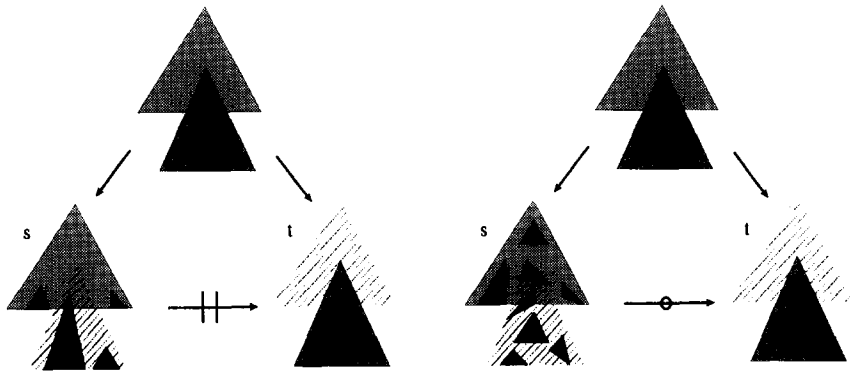


Fig. 1. Parallel closed and development closed.

(ii) If the amount is not nought, we will transform the span between the terms t and r into another span $t \leftarrow\!\!\circ\!\!\rightarrow_{\mathscr{V}'} s' \rightarrow\!\!\circ\!\!\leftarrow_{\mathscr{U}'} r$ between the same two terms, which has less interference. Consider among all the interfering steps between $\mathscr{V}$ and $\mathscr{U}$ an innermost one, say (wlog) $v \in \mathscr{V}$ and $s \rightarrow_v t'$. Because we are dealing with terms, there exists a unique step $u \in \mathscr{U}$ and $s \rightarrow_u r'$ interfering with v. By assumption all critical pairs are development closed, so there exists a simultaneous set C of redexes, such that $t' \rightarrow\!\!\circ\!\!\leftarrow_C r'$.

Claim 23. *Let* $s' =^{\text{def}} t'$, $\mathscr{V}'$ *be the residuals of* $\mathscr{V}$ *after* v (V/v *in the figure*), *and let* $\mathscr{U}'$ *be the union of* C *with the residuals of* $\mathscr{U}^-$ *after* v ($C + (U - u)/v$ *in the figure*), *where* $\mathscr{U}^- =^{\text{def}} \mathscr{U} - \{u\}$. *Then* $t \leftarrow\!\!\circ\!\!\rightarrow_{\mathscr{V}'} s' \rightarrow\!\!\circ\!\!\leftarrow_{\mathscr{U}'} r$, $\mathscr{V}'$ *and* $\mathscr{U}'$ *are simultaneous sets of redexes and the diamond property for the span has become less difficult.* (*see Fig.* 2).

The formal proof of the claim needs some sophisticated manipulation with descendants ([41, Lemma 12]), but the intuition is simple: an 'innermost' critical span $\leftarrow_v;\rightarrow_u$ is 'replaced' by a development step $\rightarrow\!\!\circ\!\!\leftarrow_C$, yielding $t \leftarrow\!\!\circ\!\!\rightarrow_{\mathscr{V}'} s' \rightarrow\!\!\circ\!\!\leftarrow_C r' \rightarrow\!\!\circ\!\!\leftarrow_{\mathscr{U}-\{u\}} r$. Since u was the only step in $\mathscr{U}$ which interfered with v, steps in C must be simultaneous with (the residuals after v of) the steps in $\mathscr{U} - \{u\}$ and this yields the development span $t \leftarrow\!\!\circ\!\!\rightarrow_{\mathscr{V}'} s' \rightarrow\!\!\circ\!\!\leftarrow_{\mathscr{U}'} r$. To prove that the span has become less difficult, one observes that by innermostness of v, the amount of interference can have changed (decreased) only for the steps in $\mathscr{V}$ interfering with u, and that it has actually decreased for the step v. □

From Lemmata 3 and 8 we have:

Corollary 24. *Let* $\mathscr{H}$ *be a left-linear development closed PRS, then* $\mathscr{H}$ *is confluent.*

Let us remark that Corollary 24 does extend the result that weakly orthogonal PRSs are confluent [45], but the proof here uses the term structure in an essential way.

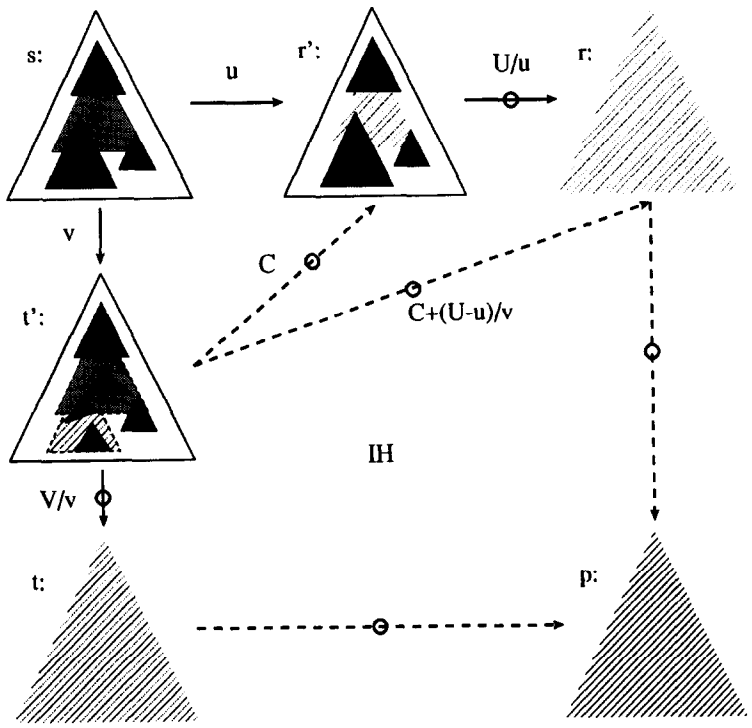


Fig. 2. Proof of the claim (only patterns of steps having overlap with u are shown).

Example 25. Consider the following variant from [37, Example 4.9] of the lambda calculus of [6, Definition 14.3.1]. To the lambda calculus with beta and eta rule, a constant $\bot: o$ representing the completely undefined term, is added, together with the two rewrite rules

$$\mathsf{app}(\bot, X) \to_{\bot\mathsf{app}} \bot$$
$$\mathsf{abs}(x.\bot) \to_{\bot\mathsf{abs}} \bot$$

This system has (on top of the trivial ones between beta and eta) two critical spans:

$$\begin{array}{ccc} & \mathsf{abs}(x.\bot) & \\ {\scriptstyle \nearrow \bot\mathsf{app}} & & \\ \mathsf{abs}(x.\mathsf{app}(\bot, x)) & \downarrow \bot\mathsf{abs} & \\ {\scriptstyle \searrow \mathsf{eta}} & & \\ & \bot & \end{array} \qquad \begin{array}{ccc} & \mathsf{app}(\bot, Y) & \\ {\scriptstyle \nearrow \bot\mathsf{abs}} & & \\ \mathsf{app}(\mathsf{abs}(x.\bot), Y) & \downarrow \bot\mathsf{app} & \\ {\scriptstyle \searrow \mathsf{beta}} & & \\ & \bot & \end{array}$$

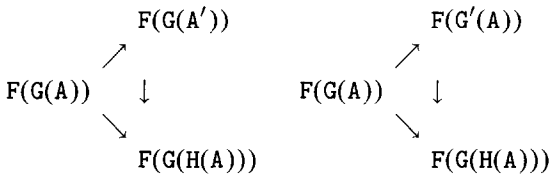
both of which are development closed, hence this PRS is confluent by Corollary 24.

The next example shows that by weakening orthogonality to development closedness, the proof invariant for confluence had to be weakened from the triangle to the diamond property.

Remark 26. Consider the following TRS adapted from [48, pp. 158, 159].

$$\begin{aligned} \mathtt{A} &\to \mathtt{A}' \\ \mathtt{A}' &\to \mathtt{H(A)} \\ \mathtt{G}(X) &\to \mathtt{G}'(X) \\ \mathtt{G}'(X) &\to \mathtt{H(G}(X)) \\ \mathtt{F(G(A))} &\to \mathtt{F(G(H(A)))} \end{aligned}$$

It has two development (even parallel) closed critical pairs:



One easily checks that starting from the term $\mathtt{F(G(A))}$, $\multimap$ does not have the triangle property (consider all five possible developments).

5.2. *Toyama*

The main result of [54] is an improvement of the result of the previous subsection by weakening the condition on a critical pair (s,t) in case of overlay to the existence of a term r such that $s \multimap\!\!\!\!\!\!+\; r \twoheadleftarrow t$. Analogous to the above extension, this result can be extended to only requiring $s \multimap r \twoheadleftarrow t$ in case of overlay.

Obviously, the diamond propery will fail to hold in general, and we have to weaken our proof invariant further to strong confluence.

Lemma 27. *Let $\mathscr{H}$ be a left-linear* almost development closed *PRS, that is,*

(i) $s \multimap t$, *or*

(ii) (s,t) *is an overlay critical pair and* $s \multimap ; \twoheadleftarrow t$,

for every critical pair (s,t); *then* $\multimap$ *is strongly confluent.*

Proof. The proof is a modification of the proof of Lemma 22, now showing that every development span can be made into a strong confluence diagram, instead of a diamond. Again we explain only the idea and refer the reader to [41, Lemma 14] for the details. The idea is not to take symbols taking part in overlays between the development steps into account, for the amount of interference. This changes nothing in the second (induction) part of the proof of Lemma 22. The base case has become more difficult now, since we are possibly left with a number of overlays between the sets of redexes in the development span. Strong confluence can be proven in that case, by induction on the size of $\mathscr{U}$ in the span $t \multimap_{\mathscr{V}} s \multimap_{\mathscr{U}} r$, making use of the fact that taking an innermost step $u \in \mathscr{U}$ decreases the size. □

From Lemmata 3 and 8 we have:

Corollary 28. *Let $\mathscr{H}$ be a left-linear almost development closed PRS, then $\mathscr{H}$ is confluent.*

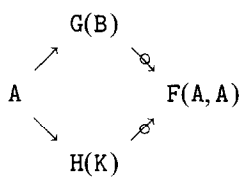
Thus we are generalising Toyama's [54, Corollary 3.2] results. Alas, except for rewriting systems which are already weakly orthogonal, there do not seem to be (m)any natural almost development closed PRSs.

Example 29. Confluence of the following rewriting systems is a consequence of Corollary 28.

(i) Consider the (non-orthogonal, non-terminating, non-right-linear) (and non-natural) TRS having rules

$$\begin{aligned} \mathtt{A} &\to \mathtt{G(B)} \\ \mathtt{A} &\to \mathtt{H(K)} \\ \mathtt{G}(X) &\to \mathtt{F}(X,X) \\ \mathtt{H}(X) &\to \mathtt{F}(X,X) \\ \mathtt{B} &\to \mathtt{A} \\ \mathtt{K} &\to \mathtt{A} \end{aligned}$$

There are two overlay critical pairs, and almost development closedness for both of them is checked by



(ii) Consider the TRS having rules

$$\begin{aligned} \mathtt{+}(X,\mathtt{0}) &\to \mathtt{0} \\ \mathtt{+}(\mathtt{0},X) &\to \mathtt{0} \\ \mathtt{+}(\mathtt{S}(X),Y) &\to \mathtt{S}(\mathtt{+}(X,Y)) \\ \mathtt{+}(Y,\mathtt{S}(X)) &\to \mathtt{S}(\mathtt{+}(X,Y)) \end{aligned}$$

One easily checks that all critical pairs are overlays and satisfy the second clause of the definition of almost development closedness.

(iii) An example to which the corollary also applies (but not immediately since the system is not a PRS because the pattern condition is violated) is formed by the

following definitional expansion rules:

$$\mathsf{app}(\mathsf{abs}(x.X(x,x)),Y) \to_{\Delta 2} \mathsf{app}(\mathsf{abs}(x.X(x,Y)),Y)$$
$$\mathsf{app}(\mathsf{abs}(x.X),Y) \to_{\Delta 0} X$$

In a more human-friendly format these rules read

$$\mathsf{let}\ x = Y\ \mathsf{in}\ X(x,x) \to_{\Delta 2} \mathsf{let}\ x = Y\ \mathsf{in}\ X(x,Y)$$
$$\mathsf{let}\ x = Y\ \mathsf{in}\ X \to_{\Delta 0} X$$

The first rule allows to expand some abbreviations x to their definition Y in the text X, e.g. $\mathsf{let}\ x = 2\ \mathsf{in}\ x \times x =^? x + x \to_{\Delta 2} \mathsf{let}\ x = 2\ \mathsf{in}\ x \times 2 =^? 2 + x$. This step is possible, because by writing $X(x,x)$ we specify that when applying this rule, each of the x's may refer to arbitrarily many occurrences of x in X.[7] The second rule allows to forget the fact that x is an abbreviation for Y, when it is not used in X anyway, e.g. $\mathsf{let}\ x = 2\ \mathsf{in}\ 1 + 28 \to_{\Delta 2} 1 + 28$. Note that since X does not have x as argument, this rule is not applicable when x occurs in X, just like in the eta rule. All critical pairs are clearly overlays, and the only interesting overlay is between $\Delta 2$ and itself (in a proper PRS an overlay critical pair between a rule and itself is always trivial), where one step expands some occurrences of an abbreviation and the other step some others. Observing that one can expand the union of these two sets is sufficient to yield almost development closedness.

6. Conclusion

We have presented three adaptations of the confluence by orthogonality method. We have shown how these, mostly minor, modifications can lead to confluence results for rewriting systems which are sufficiently close to orthogonal rewriting systems. Further variations are certainly possible. For one, we have performed our work in a more general setting (see [43, 48]) making methods available not only to (first- and higher-order) term rewriting systems, but also to net and graph rewriting systems. The first goal is then to get a confluence by orthogonality result and once this is established, one can vary on the proof method again. For example, we would like to establish confluence results for e.g. weakly orthogonal graph rewriting systems, or development closed term graph rewriting systems.

Also many variations on the theme of 'confluence via conditions on critical pairs' are possible, e.g. for first-order TRSs, confluence results have been obtained [53, 23] for TRSs satisfying certain closure conditions on *parallel critical pairs*, where most general ways of interference between one left-hand side and a number of parallel redexes (instead of just one) are considered. A natural extension is then to consider

[7] It is not possible to specify a rewrite rule which only allows to expand one abbreviation at a time. For this, types more expressive than simple types are needed, e.g. linear logic formulae.

simultaneous critical pairs, i.e. most general ways of interference between left-hand sides and sets of simultaneous redexes. Still other variations would arise from looking at PRSs having *partial development closed* (as suggested by Aart Middeldorp) or *superdevelopment closed* critical pairs. (A partial development is a prefix of a serialisation of a development, and a superdevelopment is a rewrite sequence in which only 'upward created' redexes may be contracted [47].)

Acknowledgements

I thank Femke van Raamsdonk for collaboration on Higher-Order Rewriting Systems and I acknowledge pleasant and useful discussions with Yohji Akama, Roberto Di Cosmo, Bernhard Gramlich, Stefan Kahrs, Delia Kesner, Jan Willem Klop, Aart Middeldorp, Paul-André Melliès, Tobias Nipkow, Mizuhito Ogawa, Masako Takahashi, Yoshihito Toyama, and Fer-Jan de Vries. All errors and misinterpretations are mine.

References

[1] S. Abramsky, D.M. Gabbay and T.S.E. Maibaum, eds., *Handbook of Logic in Computer Science*, Vol. 2, *Background: Computational Structures* (Oxford Univ. Press, New York, 1992).

[2] P. Aczel, A general Church–Rosser theorem, Tech. Report, University of Manchester, July 1978.

[3] Y. Akama, On Mints' reduction for ccc-calculus, in: [8, pp. 1–12].

[4] A. Asperti and C. Laneve, Interaction systems I: the theory of optimal reductions, *Math. Struct. Comput. Sci.* **4** (1994) 457–504.

[5] A. Asperti and C. Laneve, Interaction systems II: the practice of optimal reductions, *Theoret. Comput. Sci.* **159** (1996) 191–244.

[6] H.P. Barendregt, *The Lambda Calculus, its Syntax and Semantics*, Studies in Logic and the Foundations of Mathematics Vol. 103 (Elsevier, Amsterdam, revised edition, 1984).

[7] H.P. Barendregt, Lambda calculi with types, in: [1, pp. 117–310].

[8] M. Bezem and J.F. Groote, ed., in: *Proc. Internat. Conf. on Typed Lambda Calculi and Applications, TLCA'93, March 1993*, Utrecht, Netherlands, Lecture Notes in Computer Science, Vol. 664 (Springer Berlin, 1993).

[9] M. Bognar, A survey of abstract rewriting, Master's Thesis, Vrije Universiteit, Amsterdam, June 1995.

[10] A. Church, A set of postulates for the foundation of logic *Ann. Math.* **33** (1932) 346–366; **34** (1933) 839–864.

[11] A. Church and J.B. Rosser, Some properties of conversion. *Trans. Amer. Math. Soc.* **39** (1936) 472–482.

[12] D. Čubrić, Embedding of a free cartesian closed category into the category of sets. On triples.math.mcgill.ca as pub/cubric/frccc.ps.gz, April 1993.

[13] H.B. Curry, Grundlagen der kombinatorischen Logik. Teil I, II. *Amer. J. Math.* LII (1930) 509–536 and 789–834.

[14] H.B. Curry and R. Feys, *Combinatory Logic*, Vol. I. Studies in Logic and the Foundations of Mathematics (North-Holland, Amsterdam, 1958).

[15] N. Dershowitz and J.-P. Jouannaud. Rewrite systems, in: J. van Leeuwen, ed., *Handbook of Theoretical Computer Science*, Vol. *B: Formal Models and Semantics* (Elsevier, Amsterdam, 1990) 243–320.

[16] R. Di Cosmo and D. Kesner, Combining first order algebraic rewriting systems, recursion and extensional lambda calculi, in: S. Abiteboul and E. Shamir, eds., *Automata, Languages and Programming, Proc. 21st Internat. Coll. ICALP'94*, Jerusalem, Israel, 11–14, July 1994, Lecture Notes in Computer Science, Vol. 820 (Springer, Berlin, 1994) 462–472.

[17] R. Di Cosmo and D. Kesner, Simulating expansions without expansions, *Math. Struct. Comput. Sci.* **4** (1994) 1–48.
[18] R. Di Cosmo and A. Piperno, Expanding extensional polymorphism, in: M. Dezani-Ciancaglini and G. Plotkin, eds., *Proc. 2nd Internat. Conf. on Typed Lambda Calculi and Applications, TLCA'95*, Edinburgh, UK, 10–12 April, 1995, Lecture Notes in Computer Science, Vol. 902 (Springer, Berlin, 1995) 139–153.
[19] G. Dowek, J. Heering, K. Meinke and B. Möller, eds., *Higher-Order Algebra, Logic, and Term Rewriting, Proc. 2nd Internat. Workshop, HOA'95*, Paderborn, Germany, September 1995, Selected Papers, Lecture Notes in Computer Science, Vol. 1074 (Springer, Berlin, 1996).
[20] A. Geser, Relative termination, Ph.D. Thesis, University of Passau, 1990. Reprint, Ulmer Informatik-Berichte Nr. 91-03, Universität Ulm.
[21] J.-Y. Girard, Y. Lafont and P. Taylor. *Proofs and Types*, Cambridge Tracts in Theoretical Computer Science, Vol. 7 (Cambridge Univ. Press, Cambridge, 1989; Cambridge, 1990 reprinting).
[22] G. Gonthier, J.-J. Lévy and P.-A. Melliès, An abstract standardisation theorem, in: *Proc. 7th Ann. IEEE Symp. on Logic in Computer Science*, Santa Cruz, CA, 22–25 June, 1992 (IEEE Computer Society Press, Los Alamitos, CA, 1992) 72–81.
[23] B. Gramlich, Confluence without termination via parallel critical pairs, in: *Proc. CAAP'96*, 1996.
[24] J.R. Hindley, The Church–Rosser property and a result in combinatory logic. Ph.D. Thesis, University of Newcastle upon Tyne, 1964.
[25] R. Hindley, Reductions of residuals are finite, *Trans. Amer. Math. Soc.* **240** (1978) 345–361.
[26] G. Huet, Confluent reductions: abstract properties and applications to term rewriting systems *J. Assoc. Comput. Machinery* **27** (1980) 797–821.
[27] G. Huet, Residual theory in λ-calculus: a formal development, *J. Functional Programming* **4** (1994) 371–394.
[28] S. Kahrs, Context rewriting, in: M. Rusinowitch and J.L. Rémy, eds., *Conditional Term Rewriting Systems, 3rd Internat. Workshop, CTRS-92*, Pont-à-Mousson, France, 8–10 July 1992, Lecture Notes in Computer Science,Vol. 656 (Springer, Berlin, 1993) 21–35.
[29] Z.O. Khasidashvili, Expression reduction systems, in: *Proc. I. Vekua Institute of Applied Mathematics*, Vol. 36, Tbilisi (1990) 200–220.
[30] Z. Khasidashvili, The Church–Rosser theorem in orthogonal combinatory reduction systems, Rapports de Recherche 1825, INRIA-Rocquencourt, December 1992.
[31] J.W. Klop, Term rewriting systems, in: [1, pp. 1–116].
[32] J.W. Klop, Combinatory reduction systems. Ph.D. Thesis, Rijksuniversiteit Utrecht, June 1980; Mathematical Centre Tracts, Vol. 127.
[33] J.W. Klop, V. van Oostrom and F. van Raamsdonk, Combinatory reduction systems, introduction and survey, *Theoret. Comput. Sci.* **121** (1993) 279–308.
[34] P.-A. Melliès, Description Abstraite des Systèmes de Réécriture, Thèse de doctorat, Université Paris VII, 1996, to appear.
[35] D. Miller, A logic programming language with lambda-abstraction, function variables, and simple unification, in: P. Schroeder-Heister, ed., *Extensions of Logic Programming, Proc. Internat. Workshop*, Tübingen, FRG, 8–10 December 1989, Lecture Notes in Artificial Intelligence, Vol. 475 (Springer, Berlin, 1991) 253–281.
[36] G.E. Mints, Closed categories and the theory of proofs, *J. Sov. Math.* **15** (1981) 45–62. Translation into English of [38, pp. 83–114].
[37] R. Mayr and T. Nipkow, Higher-order rewrite systems and their confluence. On ftp.informatik.tu-muenchen.de as local/lehrstuhl/nipkow/hrs.dvi.gz, November 1994, *Theoret. Comput. Sci.*, to appear.
[38] G.E. Mints and V.P. Orevkov, eds., *Theoretical Applications of the Method of Mathematical Logic, Part II*, Zapiski Nauchnykh Seminarov Leningradskogo Otdeleniya Matematicheskogo Instituta im. V.A. Steklova AN SSSR, Vol. 68, 1977.
[39] T. Nipkow, Orthogonal higher-order rewrite systems are confluent, in: [8, pp. 306–317].
[40] T. Nipkow, Higher-order critical pairs, in: *Proc. 6th Ann. IEEE Symp. on Logic in Computer Science*, Amsterdam, Netherlands, 15–18 July, 1991 (IEEE Computer Society Press, Los Alamitos, CA, 1991) 342–349.
[41] V. van Oostrom, Development closed critical pairs, in: [19, pp. 185–200].
[42] V. van Oostrom, Confluence for abstract and higher-order rewriting, Ph.D. Thesis, Vrije Universiteit, Amsterdam, March 1994.

[43] V. van Oostrom, Developing developments, Tech. Report ISRL-94-4, Information Science Research Laboratory, Basic Research Laboratories, Nippon Telegraph and Telephone Corporation, December 1994; extended abstract.

[44] V. van Oostrom and F. van Raamsdonk, Comparing combinatory reduction systems and higher-order rewrite systems, in: J. Heering, K. Meinke, B. Möller and T. Nipkow, eds., *Higher-Order Algebra, Logic, and Term Rewriting, Proc. 1st Internat. Workshop, HOA'93*, Amsterdam, Netherlands, 23–24 September, 1993, Selected Papers, Lecture Notes in Computer Science, Vol. 816 (Springer, Berlin, 1994) 276–304.

[45] V. van Oostrom and F. van Raamsdonk, Weak orthogonality implies confluence: the higher-order case, in: A. Nerode and Yu. V. Matiyasevich, eds., *Logical Foundations of Computer Science, Proc. 3rd Internat. Symp., LFCS'94*, St. Petersburg, Russia, July 1994, Lecture Notes in Computer Science, Vol. 813 (Springer, Berlin, 1994) 379–392.

[46] C. Prehofer, *Solving higher-order equations: from logic to programming*, Ph.D. Thesis, Technische Universität München, February 1995; Report TUM-19508.

[47] F. van Raamsdonk, Confluence and superdevelopments, in: C. Kirchner, ed., *Rewriting Techniques and Applications, Proc. 5th Internat. Conf., RTA'93*, Montreal, Canada, 16–18 June, 1993, Lecture Notes in Computer Science, Vol. 690 (Springer, Berlin 1993) 168–182.

[48] F. van Raamsdonk, Confluence and normalisation for higher-order rewriting, Ph.D. Thesis, Vrije Universiteit, Amsterdam, May 1996.

[49] B.K. Rosen, Tree-manipulating systems and Church–Rosser theorems, *J. Assoc. Comput. Machinery* **20** (1973) 160–187.

[50] D.E. Schroer, The Church-Rosser theorem, Ph.D. Thesis, Cornell University, Ithaca, NY, 1965.

[51] J.G. Stell, Modelling term rewriting systems by sesqui-categories, Tech. Report TR94-02, Keele University, June 1994.

[52] M. Takahashi, λ-calculi with conditional rules, in: [8, pp. 406–417].

[53] Y. Toyama, On the Church–Rosser property of term rewriting systems, Tech. Report 17672, NTT ECL, December 1981 (in Japanese).

[54] Y. Toyama, Commutativity of term rewriting systems, in: F. Fuchi and L. Kott, eds., *Programming of Future Generation Computer*, Vol. II (North-Holland, Amsterdam, 1988) 393–407.

[55] D.A. Wolfram, *The Clausal Theory of Types*, Cambridge Tracts in Theoretical Computer Science, Vol. 21 (Cambridge Univ. Press, Cambridge, 1993).