# Developing Developments, Draft

Vincent van Oostrom

NTT Basic Research Laboratory
Information Processing Principles Research Group
3–1, Morinosato Wakamiya, Atsugi-Shi
Kanagawa Pref., 243-01, Japan
oostrom@theory.ntt.jp

28 September, 1994

**Abstract**

In this note it is shown how the method of confluence by developments for orthogonal rewriting systems can be adapted in some ways to obtain confluence for 'almost' orthogonal rewrite systems.

## 1. Introduction

We think the proofmethod of confluence by developments has not been fully exploited, leaving ample room for further development. In this note, we show how this method, usually employed to prove confluence of orthogonal rewriting systems ([CR36, CF58, Klo80, Oos94]), can be adapted in some easy ways to prove confluence in case of 'almost' orthogonal rewriting systems. In Section 2 we revisit the main ingredients of the confluence by developments method: stripping, ragging and orthogonality. There's no Section 3. In Section 4 we show how the proofmethod can be adapted to deal with weakly orthogonal systems. In Section 5 the proofmethod is further adapted to show confluence in the case of lambda calculus with `beta`, `eta` and `omega`-rules. This system is not even weakly orthogonal, but still developments can be used to prove its confluence. Finally, in Section 6 the method is adapted in order to deal with typed lambda calculi with expansion rules. Such calculi have received considerable attention lately. Sections 4, 5 and 6 are independent from each other.

Remark 1.1. We assume the reader to be familiar with rewriting, developments, and the concept of orthogonality.

## 2. Confluence by Developments

In this section, we informally recapitulate the main ingredients of the confluence by developments method. For standard rewriting no(ta)tions we refer to [DJ90] and [Klo92]. For ones which cannot be found there to [Oos94].

### 2.1. Strip

Confluence of a rewriting system $\to$ follows from the diagram

$$\begin{tikzcd} a \arrow[r, twoheadrightarrow] \arrow[d] & a' \arrow[d, dashed, twoheadrightarrow] \\ b \arrow[r, dashed, twoheadrightarrow] & b' \end{tikzcd}$$

by induction on the length of the expansion in a divergence $\twoheadleftarrow ; \twoheadrightarrow$. This diagram was dubbed *strip lemma* by Barendregt ([Bar84, p. 279]), because of its association to stripping a woman.

## 2.2. Rag

The strip lemma follows from the diagram

$$\begin{tikzcd} a \arrow[r] \arrow[d, two heads, "P"'] & a' \arrow[d, dashed, two heads, "P"'] \\ b \arrow[r, dashed, two heads] & b' \end{tikzcd}$$

by induction on the length of the reduction in a divergence $\twoheadleftarrow ; \twoheadrightarrow_P$, where $P$ is some property of rewrites such that $P(\rightarrow)$, i.e. every one step rewrite satisfies the property (needed to start the induction). This diagram will be called the *rag lemma*. The problem of proving confluence is reduced in this way to finding a property $P$, such that the rag lemma holds for $P$.

## 2.3. Orthogonality

The notion of *orthogonality* in rewriting systems allows for the definition of a property $P$, such that the rag lemma holds. In order to understand orthogonality, it might be helpful to think of rewrite steps as actions operating on parts of an initial object resulting in a final object, giving rise to three obvious notions:

1. the part in the initial object on which an action operates is said to be *erased* (by that action),
2. the part in the final object which is a result from performing the action is said to be *created*,
3. the part which is not affected by the action is called the surrounding part. The surrounding part in the final object is said to *descend* from the surrounding part in the initial object (in fact, steps may erase or duplicate parts of its surroundings but only in a logical way, that is, without looking at how these parts are constructed).

Using these intuitive notions, orthogonality is defined to consist of the following three ingredients:

1. Distinct actions in the set act on distinct parts (*parallelism*),
2. every development is finite (*finite developments*) (a *development* is a rewrite sequence in which only actions operating on parts descending from parts in the initial object are performed),
3. the order in which distinct actions are performed does not influence the effect on their common surroudings (*parametricity*).

For formal definitions of orthogonality, see e.g. [GLM92, Oos94].

The point of orthogonality is that it allows for a proof of the rag lemma by taking $P(d) =^{\mathrm{def}}$ *d is a complete development of some set of redexes* (a development is *complete* if it cannot be extended), since one can construct the diagram

$$\begin{tikzcd} a \arrow[rrr, "u"'] \arrow[ddd, two heads, "d"] & & & a' \arrow[ddd, dashed, two heads] \\ & \mathcal{V} \cup \{u\} \arrow[r, dotted, "u"'] \arrow[d, dotted, "\mathcal{V}"] & \{\mathcal{V} \lfloor u \rfloor\} \arrow[d, dotted, "\{\mathcal{V} \lfloor u \rfloor\}"] & \\ & \{u \lfloor d \rfloor\} \arrow[r, dotted, "\{u \lfloor d \rfloor\}"'] & \emptyset & \\ b \arrow[rrr, dashed, two heads] & & & b' \end{tikzcd}$$

where $\mathcal{V} \cup \{u\}$ is a set of (parallel) rewrite steps acting on $a$. The labels on the dotted lines indicate which set of steps is developed in the rewrite on its outside. Here $\{\mathcal{U}\lfloor e\rfloor\}$ denotes the set of descendants of a set $\mathcal{U}$ after performing a rewrite $e$. The basic idea is that one can construct the object $b'$ from $a$ by performing the set of steps $\mathcal{V} \cup \{u\}$ 'in parallel', because the steps are orthogonal to each other. The diagram is known as the *parallel moves lemma* (see Curry and Feys [CF58, p. 113]).

Two points are worth noting about the proof of the parallel moves lemma.

1. Given that every development is finite, it is sufficient to assume that orthogonality holds for every pair of rewrite steps (*squaring*). Then it is shown that $\mathcal{V} \cup \{u\}$ is orthogonal, by induction on the size of $\mathcal{V}$ using squaring.

2. Any tail of a complete development is a complete development. In particular, the set of descendants of a parallel set is parallel again (*preservation of parallelism*).

## 4. Weak Orthogonality

It is sometimes easy to adapt the proof of confluence by developments to deal with rewriting systems which are not quite orthogonal. In this section we show that's the case for *weakly orthogonal* systems, where distinct actions are allowed to operate on the same part of an object (giving rise to so called *critical pairs*), but the following condition is imposed

> if two rewrite steps act on the same part, then they have the same result (*trivial critical pairs*),

We show that the rag lemma holds using the same property $P$ as in the preceding section, by making an easy case distinction.

1. If $u$ acts on some part on which some step, say $v$ in $\mathcal{V}$ acts as well, then one can construct the diagram

$$\begin{tikzcd}
a \arrow[rrr, "u"'] \arrow[ddd, two heads, "d"] & & & a' \arrow[ddd, dashed, two heads] \\
& \mathcal{V} \arrow[r, dotted, "v"'] \arrow[d, dotted, "\mathcal{V}"] & \{\mathcal{V}\lfloor v\rfloor\} \arrow[d, dotted, "\{\mathcal{V}\lfloor v\rfloor\}"] & \\
& \emptyset \arrow[r, dotted, "\emptyset"'] & \emptyset & \\
b \arrow[rrr, dashed, two heads] & & & b = b'
\end{tikzcd}$$

   Note that the step $u$ is performed by developing the step $v$. By the weak orthogonality assumption these two steps result in the same object $a'$.

2. If $u$ acts on a part disjoint from the parts on which $\mathcal{V}$ acts, then one can use exactly the same diagram as in the previous section on orthogonality.

This proof (which can be found in more detail in [Oos94, OR94]) is based on preservation of parallelism as in the orthogonal case, but squaring is replaced by the condition:

> Every triple of pairwise orthogonal steps is orthogonal (*cubing*). Then it is shown by induction on the size of $\mathcal{V}$ that $\mathcal{V} \cup \{u\}$ is orthogonal if $\mathcal{V}$ is so and $u$ is pairwise orthogonal to every step in $\mathcal{V}$.

# 5. Beta-Eta-Omega

Although we do not have an axiomatisation of a notion of 'almost' orthogonality, some further 'almost' orthogonal rewriting systems are not difficult to find. In this section we adapt the confluence via developments method to show confluence of the rewriting system $\mathcal{BHO}$ (see [Bar84]). The terms are untyped lambda terms and the rules are:

$$\begin{array}{rcl} (\lambda x.sx)t & \to_\beta & st \\ \lambda x.(s)x & \to_\eta & s \text{ , if } x \text{ does not occur in } s \\ s & \to_\Omega & \Omega \text{ , if } s \text{ is unsolvable and } s \not\equiv \Omega \end{array}$$

In the case of weak orthogonality the confluence proof was based on the idea that a case of overlap could be *simulated* by an orthogonal set of rewrite steps (by just choosing one of the steps). Here we adapt this idea somewhat: a case of overlap may be *extended* by an orthogonal set of rewrite steps, i.e. there should exist an orthogonal set of rewrite steps being able to do at least as much as each of the overlapping steps separately.

Informally, since simulation is a special case of extension, the weak overlap between $\beta$ and $\eta$ does not cause trouble. However, there are also the conflicts arising from $(\lambda x.s)t$, where $\lambda x.s$ is unsolvable, and from $\lambda x.(s)x$, where $(s)x$ is unsolvable. The key to confluence is that in both cases the two overlapping steps can be extended by the step rewriting the initial term directly to $\Omega$.

Formally, consider the divergence:

$$\begin{tikzcd} a \arrow[r, "u"'] \arrow[d, "\mathcal{V}"] & c \\ b & \end{tikzcd}$$

The following situations can occur.

1. $u$ is orthogonal to $\mathcal{V}$. This is trivial.

2. $u$ has overlap with some (possibly more than one) rewrite step $v \in \mathcal{V}$. There are three cases to consider.

   (a) In case of overlap between an $\eta$- and a $\beta$-step we procede as in the case of weak orthogonality.

   (b) An overlap between a $\beta$- and an $\Omega$-step is dealt with by the diagram

$$\begin{tikzcd} (\lambda x.s)t \arrow[r, "\Omega"'] \arrow[d, "\beta"] \arrow[dr, "\Omega"'] & (\Omega)t \arrow[d, "\Omega"] \\ s[x := t] \arrow[r, "\Omega"'] & \Omega \end{tikzcd}$$

   The diagonal $\Omega$-redex (which exists by definition of unsolvability, see [Bar84]) is added to $\mathcal{V}$. This $\Omega$-redex is preserved by both of the other two initial steps, and the right and bottom $\Omega$-steps are its descendants along these initial steps. To show that this step is orthogonal to $\mathcal{V}$, we distinguish two cases:

   i. If $u$ is the $\Omega$-step, it follows from the orthogonality of $\mathcal{V}$,

   ii. If $u$ is the $\beta$-step, it follows because an overlap with an $\eta$-step would only be possible if the $\beta$- and $\eta$-step already did have overlap.

(c) An overlap between an $\Omega$- and an $\eta$-step is dealt with by the diagram

$$\begin{tikzcd}
\lambda x.(s)x \arrow[r, "\Omega"'] \arrow[d, "\eta"] \arrow[dr, "\Omega"'] & \lambda x.\Omega \arrow[d, "\Omega"] \\
s \arrow[r, "\Omega"'] & \Omega
\end{tikzcd}$$

using reasoning completely analogous to the preceding case.

There are two small difficulties in this approach, which can easily be overcome.

1. $\Omega$-redexes are not head defined (in the sense of [Oos94], i.e. there's no specific (head-)symbol in a term representing its unsolvability), but subterm defined. Tracing edges (the ones connecting subterms which are $\Omega$-redexes) works as well.

2. $\Omega$-redexes are not preserved by rewriting if one just looks at subterms, because of the side-condition ($s \not\equiv \Omega$) on the rule. However, omitting this condition does not influence confluence.

Having noticed this, the Finite Developments Theorem is easily proved using standard techniques, so confluence follows.

# 6. Expansion

In this section, we use developments to prove confluence of some typed lambda calculi with expansion rules. Recently, 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. That is, the rules of a typed lambda calculus $\mathcal{H}$ are split into a set $\mathcal{R}$ of reduction rules (e.g. $\beta$) and a set $\mathcal{E}$ of expansion rules (e.g. $\eta^{-1}$). For example, one can take the following set of rules ([DCK93b]).

$$\begin{array}{rll}
(\lambda x.s)t & \to_\beta & s[x := t] \\
s & \to_{\eta^{-1}} & \lambda x.(s)x \text{ , if } x \text{ does not occur in } s \\
\pi_1\langle s,t\rangle & \to_{\pi_1} & s \\
\pi_2\langle s,t\rangle & \to_{\pi_2} & t \\
s & \to_{\pi^{-1}} & \langle \pi_1 s, \pi_2 s\rangle \\
s & \to_{\tau^{-1}} & *
\end{array}$$

These are not all the rules considered in the literature but they will suffice to illustrate our method. The following remarks are in order:

1. There are three types of constructors: $\lambda$ the function type constructor, $\pi$ the ordered product type constructor and $*$ the terminal type constructor.

2. For each of the types we have ways to construct terms of that type via expansion rules in $\mathcal{E}$. The expansion rules corresponding to the types of the previous item are $\eta^{-1}$, $\pi^{-1}$ and $\tau^{-1}$, respectively. Expansion rules can be thought of as inserting a so-called constructor–destructor(s) pair (in the logical meaning, i.e. corresponding to introduction and elimination) of the appropriate type at some edge in the tree representation of a term. Graphically: $- \to_{\mathcal{E}} -c\text{–}d-$. Trivial rewrite cycles would arise from this expansion in the following situations:

   (a) If the subterm at the insertion point was already in constructor format. Graphically: $-c- \to_{\mathcal{E}} -c\text{–}d\text{–}c- \to_{\mathcal{R}} -c-$.

(b) If in the original term, a destructor was applied to the insertion point. Graphically: $-d- \to_{\mathcal{E}} -d\text{–}c\text{–}d- \to_{\mathcal{R}} -d-$.

The usual side conditions on the expansion rules exactly disallow these two situations, giving rise to the *restricted* expansion rules $\overline{\mathcal{E}}$, i.e. $\overline{\eta}$, $\overline{\pi}$ and $\overline{\tau}$. As noted by Čubrić, a somewhat remarkable fact is that just disallowing these two trivial cycles entails strong normalisation.

3. Each type can be destructed via reduction rules in $\mathcal{R}$. Reduction rules can be thought of as eliminating a so-called destructor–constructor pair in a term. The function type has infinitely many ways to destruct it ($\beta$), the product type has two ways to destruct it ($\pi_1$ and $\pi_2$) and the terminal type has no way to destruct it.

The expansion rules can be viewed as bringing a term into the form required by its type. Because the terms are typed, there is no harm in doing this, since the typing already guarantees that the result of the term will be of that form eventually. This point of view was first taken by Mints ([Min81]) and further developed by several authors ([Aka93, DCK93b, Čub93]). For this approach to be correct completeness, i.e. confluence and strong normalisation, of the system must be proved.

## 6.1. A Modular Approach to Completeness

As noted in the literature, completeness can be shown most easily in a modular fashion, that is, by first proving completeness for the restricted expansions and reductions separately and then combining these results to show completeness of the complete system.

**Modularity of Confluence** Confluence can be reduced to confluence of reductions and restricted expansions separately by the Lemma of Hindley–Rosen, once it is known that restricted expansions commute with reductions, i.e. if the following diagram holds:

$$\begin{tikzcd}
s \arrow[r, twoheadrightarrow, "\mathcal{R}"'] \arrow[d, twoheadrightarrow, "\overline{\mathcal{E}}"] \arrow[dr, phantom, "(A)"] & s' \arrow[d, dashed, twoheadrightarrow, "\overline{\mathcal{E}}"] \\
t \arrow[r, dashed, twoheadrightarrow, "\mathcal{R}"'] & t'
\end{tikzcd}$$

**Modularity of Strong Normalisation** Once confluence of restricted expansion and commutation of restricted expansion and reduction is known, strong normalisation can be reduced to strong normalisation of reductions and restricted expansions separately, in case we have the following diagram:

$$\begin{tikzcd}
s_0 \arrow[r, "+", "\mathcal{R}"'] \arrow[d, twoheadrightarrow, "\overline{\mathcal{E}}"] & s_1 \arrow[r, "+", "\overline{\mathcal{E}}"'] \arrow[d, twoheadrightarrow, "\overline{\mathcal{E}}"] & s_2 \arrow[r, "+", "\mathcal{R}"'] \arrow[d, twoheadrightarrow, "\overline{\mathcal{E}}"] \arrow[dr, phantom, "(B)"] & s_3 \arrow[r, "+", "\overline{\mathcal{E}}"'] \arrow[d, twoheadrightarrow, "\overline{\mathcal{E}}"] & s_4 \arrow[r, "+", "\mathcal{H}"'] \arrow[d, twoheadrightarrow, "\overline{\mathcal{E}}"] & \cdots \\
t_0 \arrow[r, dashed, "+", "\mathcal{R}"'] & t_1 \arrow[r, phantom, "\equiv"] & t_2 \arrow[r, dashed, "+", "\mathcal{R}"'] & t_3 \arrow[r, phantom, "\equiv"] & t_4 \arrow[r, dashed, "+", "\mathcal{R}"'] & \cdots
\end{tikzcd}$$

where each $t_m$ is the $\overline{\mathcal{E}}$-normal form of $s_m$. To show that this diagram holds, one needs apart from commutation also that restricted expansion cannot erase reductions (to preserve transitivity) and that reduction cannot create restricted expansions (to preserve normal forms).

## 6.2. Commutation by Developments

In this subsection we show that restricted expansions commute with reductions in the case of the typed lambda calculus introduced above. The problem with showing commutation of $\mathcal{R}$ and $\overline{\mathcal{E}}$ is that restricted expansion rules are, well, restricted and the usual ways to prove commutation are geared to unrestricted rules. This problem is tackled in different ways by different authors, which we will comment on shortly. In this subsection we adapt the confluence by developments proofmethod of the previous sections. We think this technique is the most perspicuous one. Moreover, it straightforwardly extends to other systems as will be shown in the sequel.

We show that the (commutation version of the) rag lemma holds, by taking $P(d) =^{\mathrm{def}}$ *$d$ is a complete development of some set of expansion redexes*, i.e.

$$\begin{tikzcd} s \arrow[r, twoheadrightarrow, "\mathcal{R}"] \arrow[d, "P\,\overline{\mathcal{E}}"'] \arrow[dr, phantom, "(C)"] & s' \arrow[d, dashed, "P\,\overline{\mathcal{E}}"] \\ t \arrow[r, dashed, twoheadrightarrow, "\mathcal{R}"'] & t' \end{tikzcd}$$

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 HORS in the sense of [Oos94], so we can construct the diagram

$$\begin{tikzcd} s \arrow[r, twoheadrightarrow, "\mathcal{R}"] \arrow[d, "\overline{\mathcal{E}}"'] \arrow[dr, phantom, "(D)"] & s' \arrow[d, dashed, "\mathcal{E}'"] \\ t \arrow[r, dashed, twoheadrightarrow, "\mathcal{R}"'] & t' \end{tikzcd}$$

Remark 6.1. An expansion redex is not determined by a symbol in the term (it's not *head-defined*), but rather determined by an (typed) edge in the term. Edges can be traced without problems. Alternatively, one can take a fresh unary symbol $c_u$, for every expansion step $u$, insert that symbol at the edge of the expansion redex and trace it.

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 expansion from $s'$ to $t'$ may contain some expansions which are not restricted. The reason is that a( descendant of a)n expansion step might not be allowed anymore after a reduction, due to its context-sensitivity.

Example 6.2. In the case of $\overline{\eta}$ and $\beta$, destruction can happen in three ways, corresponding to the three ways in which redexes can be created by $\beta$-reduction.

1. The expansion $(\lambda x.x)\lambda y.y \rightarrow_{\overline{\eta}} (\lambda x.\lambda z.(x)z)\lambda y.y$ expanding $x$, is destroyed by the reduction $(\lambda x.x)\lambda y.y \rightarrow_{\beta} \lambda y.y$.
2. The expansion $(\lambda x.x)\lambda y.y \rightarrow_{\overline{\eta}} \lambda z.((\lambda x.x)\lambda y.y)z$ expanding the complete initial term, is destroyed by the reduction $(\lambda x.x)\lambda y.y \rightarrow_{\beta} \lambda y.y$.
3. The expansion $(\lambda x.(x)t)s \rightarrow_{\overline{\eta}} (\lambda x.(x)t)\lambda y.(s)y$ expanding $s$, is destroyed by the reduction $(\lambda x.(x)t)s \rightarrow_{\beta} (s)t$.

However, the unrestricted commutation diagram $(D)$ can easily be transformed into the following restricted commutation diagram:

$$\begin{tikzcd} s \arrow[rr, twoheadrightarrow, "\mathcal{R}"] \arrow[d, "\overline{\mathcal{E}}"'] & \arrow[d, phantom, "(E)"] & s' \arrow[d, dashed, "\overline{\mathcal{E}}"] \\ t \arrow[r, dashed, twoheadrightarrow, "\mathcal{R}"'] & t' \arrow[r, dashed, twoheadrightarrow, "\mathcal{R}"'] & t'' \end{tikzcd}$$

To see this, observe that the unrestricted expansion from $s'$ to $t'$ can be partitioned into a restricted expansion of maximal length to some term $t''$ followed by an unrestricted expansion to $t'$. Because every unrestricted expansion from $t''$ to $t'$ is not a restricted one, one of the side conditions must be violated, hence there exists a reduction step in the reverse direction. The diagram expresses that developments of restricted expansion steps commute with reduction steps, so we are done.

Apart from finiteness of developments, in the proof use is made of the fact that unrestricted expansions which are not restricted ones, are preserved by expansions and of the fact that restricted expansions are 'almost' preserved by expansions. The only nasty detail is that distinct expansions in $s$ may descend to expansions of 'the same edge' in $s'$ along the reduction from $s$ to $s'$ (e.g. consider the first two cases in the example above). In case both expansion redexes are restricted, expanding one of them destroys the other one. The solution is simple: select a fixed one. This guarantees that the restricted expansion of maximal length constructed in the proof is actually a development.

By the Lemma of Hindley–Rosen confluence of the complete system follows from confluence of reductions and restricted expansions separately, which 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.

## 6.3. Strong Normalisation by Developments

For the modular proof of strong normalisation we want to show $(B)$. In fact this diagram does not hold for the rewriting system having all the expansion rules, because the expansion rule $\overline{\tau}$ can erase reduction steps, which might result in empty reduction steps in the bottom of the conclusion of the diagram. However, one observes that applications of the $\overline{\tau}$-rule can be postponed (the terminal type doesn't have a destructor) so do not matter for strong normalisation. Hence, we may assume wlog. that $\overline{\mathcal{E}}$ does not contain this rule. Once this rule has been removed, expansion cannot erase reduction, so it only remains to show that reduction cannot create restricted expansion.

The following, general, reasoning applies. First, typed lambda calculi represent logics which are about rearranging, copying and deleting resources, not about creating them. In other words, each edge in the result of a reduction step descends from some edge in the initial term. Second, the only way for such a descendant to become an expansion redex is if it didn't satisfy the side conditions before the step, but does satisfy them afterwards. It is easy to see that this could only happen in case the edge was inbetween the destructor–constructor pair of the reduced redex, hence was erased by the reduction step, which is absurd.

We conclude that strong normalisation of the complete system follows from strong normalisation of restricted expansion and reduction separately. Again we are not concerned with proving those properties here. Strong normalisation of reduction is non-trivial, but (variants of) the Tait-Girard reducibility method do apply. Strong normalisation of restricted expansion follows by defining an appropriate measure on terms. Alternatively, it can be easily obtained by transforming the problem into a strong normalisation problem for a term rewriting system having types as function symbols.

## 6.4. Related Work

In this subsection we compare our method with some methods known from the literature (in no specific order).

**Čubrić** Our proofs of confluence and strong normalisation are very similar to the ones given by Čubrić in [Čub93]. The only difference is that he uses *minimal* developments to show commutation ([Čub93, Lem. 4.7]), whereas we employ developments. It's a matter of taste.

For some reason, in the literature the expansion rule $\overline{\tau}$ is often grouped together with the reduction rules, not with the other expansion rules. Maybe this is because it simplifies some proofs, but it is certainly 'wrong' from a logical/categorical point of view. Categorically speaking, expansion rules corresponds to existentially quantified (unique) arrows, so the terminal rule should be one. Operationally speaking, expansion rules construct form, so also from this point of view the terminal rule should be an expansion rule. This 'wrong' partitioning of the rules makes that first normalising with $\mathcal{R}$ and then with respect to $\overline{\mathcal{E}}$ is not a normalising strategy as shown by Čubrić ([Čub93, Rem. 4.17]). In the logical partitioning both this strategy and the dual strategy (first normalising with respect to $\overline{\mathcal{E}}$ and then with respect to $\mathcal{R}$) are normalising. This is true because reduction cannot create restricted expansions as was shown above and moreover, restricted expansions cannot create reductions, because the side conditions are precisely such that this cannot happen.

**Di Cosmo and Kesner** Di Cosmo and Kesner employ a technique different from ours to show commutation of restricted expansions and reduction in [DCK93b]. There they show that the following diagram holds:

$$\begin{tikzcd} a \arrow[r, "\mathcal{R}"'] \arrow[d, "P|\overline{\mathcal{E}}"'] & a' \arrow[d, dashed, "P|\overline{\mathcal{E}}"'] \\ b \arrow[r, dashed, two heads, "\mathcal{R}"'] & b' \end{tikzcd}$$

where $P(d) =^{\mathrm{def}} d$ *is a restricted expansion to (expansion) 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. Although our (and Čubrić's) proof of commutation based on (minimal) developments gives more refined results than their method, it's again largely a matter of taste which method one prefers. We like our method better because developments are known from the literature, so there's no need to go through many tedious induction proofs.

The lambda calculus they consider includes rules for sums and bounded and unbounded recursion. To deal with these new rules, our confluence proof need not be changed at all, since the new rules are (orthogonal) reductions rules and because the commutation proof hinges on properties of expansions, not of reductions. Confluence ([DCK93b, Thm. 8.1,8.4]) follows from confluence of expansions and reductions separately, which are not difficult to show. For strong normalisation we only need to check that reduction cannot create restricted expansion, which follows by inspection of the new rules. Hence, strong normalisation ([DCK93b, Thm. 5.14]) follows from strong normalisation of expansions and reductions separately (which of course does not hold in case of unbounded recursion). Note that strong normalisation for reductions is non-trivial and can be found there ([DCK93b, Cor. 6.11, 7.7]).

In recent work ([DCK94]), the same authors investigate modularity questions. In particular they consider modularity of confluence and strong normalisation for combinations of (left-linear) algebraic rewriting systems and extensional lambda calculi. They show these properties to be modular by a reduction to modularity of the same properties for combinations of algebraic rewriting systems and (non-extensional) lambda calculi, which are already known to hold from the literature.

In our method the proof of commutation has to be extended to deal with commutation of expansions and algebraic rewrites. Because of the very special form of algebraic rules (algebraic terms are in expansion normal form and variables are required to be of base type), algebraic steps neither create nor destroy expansions, so commutation is trivial. Hence, by the Lemma of Hindley–Rosen, confluence of the combined system can be reduced to confluence of the system without expansion rules, which proves Theorem 3.12 of [DCK94].

Also due to the form of the algebraic rules, restricted expansion cannot erase algebraic redexes and algebraic steps cannot create expansion redexes, so strong normalisation is reduced to strong normalisation of the system without expansion rules, which proves Theorem 3.10 of [DCK94].

Finally, they consider modularity of confluence for combinations of (left-linear) rewriting systems and unbounded recursion, represented by the rule

$$fix \rightarrow_{fix} \lambda x.(x)(fix)x$$

where $fix$ is a fresh constant (for every type). First, they show confluence for combinations with left-linear rewriting systems without expansion rules. Such rewriting systems belong to the class of HORSs. Because $fix$ is confluent (even orthogonal) and orthogonal to all other rules (because the constants $fix$ are fresh), the general modularity of confluence result in [Oos94, OR94] entails Theorem 4.8 of [DCK94] (even without the presence of $\beta$, which is assumed there).

Next, they consider confluence for combinations with left-linear rewriting systems with expansion rules. The general modularity of confluence result cannot be used anymore, because it is not so general. In particular it cannot deal with restricted (context-sensitive) rules. However, the following lemma applies taking $\rightarrow_1 =^{\text{def}} \rightarrow_{fix}$ and $\rightarrow_2$ the rewrite relation of the left-linear rewriting system. 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 Theorem 4.11 of [DCK94]. Note that in this case the presence of $\beta$ is essential; without it confluence (commutation) does not hold.

LEMMA 6.3. *Let $\rightarrow =^{\text{def}} \rightarrow_1 \cup \rightarrow_2$. If $\rightarrow_1$ and $\rightarrow_2$ are confluent and*

$$\begin{tikzcd} \cdot \arrow[r, two heads, "2"'] \arrow[d, two heads, "1"] & \cdot \arrow[d, dashed, two heads, "1"'] \\ \cdot \arrow[r, dashed, leftrightarrow, "*", "2"'] & \cdot \end{tikzcd}$$

*then $\rightarrow$ is confluent.*

PROOF Easy diagram chase. $\odot$

**Akama** Akama proves completeness of Mints' reduction using the following lemma ([Aka93, Lem. 4]) corresponding to diagram $(B)$.

LEMMA 6.4. *If $\overline{\mathcal{E}}$ and $\mathcal{R}$ are both complete rewriting systems and*

$$a \rightarrow_{\mathcal{R}} b \implies a{\downarrow_{\overline{\mathcal{E}}}} \rightarrow^{+}_{\mathcal{R}} b{\downarrow_{\overline{\mathcal{E}}}}$$

*then $\overline{\mathcal{E}} \cup \mathcal{R}$ is complete.*

This lemma and his completeness proof are closer in spirit to the method of Di Cosmo and Kesner, sketched in the previous paragraph, than to a proof by (minimal) developments, so the same comments apply. The typed lambda calculus for which he proves confluence is the one we presented, so our results cover his results.

**Di Cosmo and Piperno** In a very recent paper [DCP94], Di Cosmo and Piperno consider completeness of system F ([GLT89]) equiped with restricted $\eta$-expansion. The reduction rules of the system $\mathcal{H}$ are $\beta$ and $\overline{\eta}$ plus the universal reduction rule $B$:

$$(\Lambda X.s)V \quad \rightarrow_B \quad s[X := V]$$

The authors identify the difference from the first-order as residing in the fact that $B$ can create $\eta$, rendering the modular approach useless. Nevertheless, they are able to show completeness, based on the observation that $\overline{\eta}$ *can always be postponed (preserving infinity of a rewrite).* Their proof is essentially based on the following commutation lemma (cf. [Ges90, p. 47]).

LEMMA 6.5. *Let $\rightarrow_0$ and $\rightarrow_1$ be abstract rewriting systems (i.e. relations). If $\rightarrow_0$ is strongly normalising and*

$$\begin{tikzcd} \cdot \arrow[r, "0"'] \arrow[d, "1"] & \cdot \arrow[d, dashed, two heads, "1"'] \\ \cdot \arrow[r, dashed, "+", "0"'] & \cdot \end{tikzcd}$$

*holds, then $\rightarrow_0$ and $\rightarrow_1$ commute.*

The rôle this lemma plays in their proof, is similar to that of the rag lemma in the proofs we have given, so a natural question is whether we can obtain their results via our method as well. The answer is yes, as we will show. Again there's no claim to originality whatsoever concerning the result, we just like the proof.

REMARK 6.6. The authors note, like we did, that proving commutation first enables a considerable shortening of existing proofs of completeness for first-order $\lambda$-calculi. To compare their and our method, it only remains to compare their and our commutation lemmata. We think that in this case they're equally easy to show.

Showing confluence of $\mathcal{H}$ is easy. Because $\overline{\eta}$ does not duplicate $\mathcal{R}$-steps ($\beta$ or $B$), one can refine the diagram $(E)$ to:

$$\begin{tikzcd}
s \arrow[rr, "\mathcal{R}"'] \arrow[d, "\overline{\eta}"] & (F) & s' \arrow[d, dashed, "\overline{\eta}"] \\
t \arrow[r, dashed, "\mathcal{R}"'] & t' \arrow[r, dashed, two heads, "\beta"'] & t''
\end{tikzcd}$$

where the bottom $\mathcal{R}$-step descends from the top one as indicated, and all other steps in the conclusion descend from the $\overline{\eta}$-steps. By the Lemma of Hindley-Rosen, we are done using confluence of $\overline{\eta}$ and $\mathcal{R}$, both of which are known ($\beta$ and $B$ are even orthogonal to each other).

Strong normalisation is a different story. As noticed by Di Cosmo and Piperno, the modular approach fails, but $\overline{\eta}$-postponement works.

THEOREM 6.7. *If $d\colon a \to^{\omega} \ldots$ is an infinite rewrite sequence, then there exists an infinite rewrite sequence $e\colon a \to_{\mathcal{R}} ; \to^{\omega} \ldots$.*

PROOF If $d$ starts with a reduction step this is trivial, otherwise we can 'prepone' a reduction step as indicated in the following diagram

$$\begin{tikzcd}
a \arrow[r, "m", "\overline{\eta}"'] & {} \arrow[r, "\overline{\eta}"'] \arrow[d, dashed, "\mathcal{R}"] & {} \arrow[r, "\mathcal{R}"'] & {} \arrow[r, "\omega", "\mathcal{H}"'] \arrow[d, dashed, "\beta"] & {} \\
 & {} \arrow[rr, dashed, two heads, "\overline{\eta}"'] & (G) & {} \arrow[r, dashed, "\omega", "\mathcal{H}"'] & {}
\end{tikzcd}$$

Clearly, if we can show $(G)$ and $(H)$, then we are done by repeatedly preponing the reduction step until it becomes the initial step. The first one, $(G)$ is no problem, since expansion can neither create nor duplicate reduction steps so the reduction step can be preponed and the diagram follows directly from $(F)$. Since we know the system is confluent, the only problem in showing $(H)$ is preservation of infinity. For general $\beta$-reduction, this is not clear since a $\beta$-step might erase arbitrary complex subterms (viz $\mathtt{K}s$). Fortunately, we are dealing with $\beta$-steps which are *descendants of $\overline{\eta}$-steps*, in the left-hand side of $(H)$. These do not exhibit such eradicating behaviour. First note that initially, since the $\beta$-steps 'are' expansions violating (one of) the side-conditions, they must involve either of the marked symbols of a term $\lambda^{*}x.(s)^{*}x$. (We use marks as a notational device in the presentation, they have *no* formal meaning.) One observes that by performing $\mathcal{H}$-steps, such a term can only evolve to a term of the form $\lambda^{*}x.C[(s')^{*}r]$, such that

1. $x$ does not occur in the function of $(\,)^{*}$ and occurs once in its argument,
2. both the context ($C[\,]$) surrounding $(\,)^{*}$ and its argument ($r$) are expansions of a variable.

Inspired by Di Cosmo and Piperno, we define $\beta^{\star}$ as a restricted form of $\beta$-reduction, accordingly (satisfying the two conditions above):

$$\begin{aligned}
(\lambda^{*}x.C[(s)^{*}r])t &\to_{\beta^{\star}} C[(s)r[x:=t]] \\
\lambda^{*}x.C[(\lambda y.s)^{*}r] &\to_{\beta^{\star}} \lambda x.C[s[y:=r]]
\end{aligned}$$

Claim $\beta^\star\overline{\eta}$ commutes with $\mathcal{H}$ and $B$-steps are preserved in this process.
This claim settles $(H)$ and the proof, because the lefthand side of $(H)$ is a $\beta^\star$-reduction, and the top rewrite contains infinitely many $B$-steps.

Proof of claim We will first show confluence of $\beta^\star\overline{\eta}$. By Newman's Lemma and strong normalisation of $\beta^\star\overline{\eta}$ it suffices to prove local confluence. This is trivial except in case of an overlap between the two $\beta^\star$-rules:

$$\begin{tikzcd}
(\lambda^* x.C[(\lambda y.s)^* r])t \arrow[r, "\beta^\star"'] \arrow[d, "\beta^\star"] & (\lambda x.C[s[y := r]])t \arrow[d, dashed, two heads, "\eta^{-1}"] \\
C[(\lambda y.s)r[x := t]] \arrow[r, dashed, two heads, "\eta^{-1}"'] & C[(\lambda x.C[s[y := r]])r[x := t]]
\end{tikzcd}$$

A common reduct can indeed be reached by (unrestricted) expansions:

$$\begin{array}{rcl}
t & \twoheadrightarrow_{\eta^{-1}} & r[x := t] \\
(\lambda x.C[s[y := r]])t & \twoheadrightarrow_{\eta^{-1}} & C[(\lambda x.C[s[y := r]])t] \\
s & \twoheadrightarrow_{\eta^{-1}} & s[y := r[x := y]] \\
s & \twoheadrightarrow_{\eta^{-1}} & C[s]
\end{array}$$

the first two from top to bottom, and the last two from left to right. Now note that every unrestricted expansion $\twoheadrightarrow_{\eta^{-1}}$ can be transformed into a valley of the form $\twoheadrightarrow_{\overline{\eta}}\,;\,\twoheadleftarrow_{\beta^\star}$ ($(\eta^{-1}-\overline{\eta})$-steps are ${\beta^\star}^{-1}$-steps, which commute with $\overline{\eta}$ by $(F)$). Since the 'vertical' expansion is independent from the 'horizontal' one, the two corresponding valleys can be put 'in parallel' (giving a valley with a $\beta^\star$- and an $\overline{\eta}$-rewrite on each side), finally giving the common reduct.

The only remaining thing is to show that $\mathcal{R}$ commutes with $\beta^\star\overline{\eta}$. This follows from the diagram (shown by induction on the length of the lefthand side):

$$\begin{tikzcd}
\cdot \arrow[rrr, "\mathcal{R}"] \arrow[d, "\beta^\star\overline{\eta}"'] \arrow[rrrd, phantom, "(I)"] & & & \cdot \arrow[d, dashed, two heads, "\beta^\star\overline{\eta}"] & & \cdot \arrow[r, "\mathcal{R}"] \arrow[d, "\beta^\star"'] \arrow[rd, phantom, "(J)"] & \cdot \arrow[d, dashed, two heads, "\beta^\star"] \\
\cdot \arrow[rr, dashed, "\mathcal{R}"] \arrow[d, "\beta^\star\overline{\eta}"'] \arrow[rrd, phantom, "IH"] & & \cdot \arrow[r, dashed, two heads, "\beta^\star"] \arrow[d, dashed, two heads, "\beta^\star\overline{\eta}"'] \arrow[rd, phantom, "CR"] & \cdot \arrow[d, dashed, two heads, "\beta^\star\overline{\eta}"] & & \cdot \arrow[r, dashed, "\mathcal{R}"'] & \cdot \\
\cdot \arrow[r, dashed, "\mathcal{R}"'] & \cdot \arrow[r, dashed, two heads, "\beta^\star\overline{\eta}"'] & \cdot \arrow[r, dashed, two heads, "\beta^\star\overline{\eta}"'] & \cdot & & &
\end{tikzcd}$$

where $(I)$ follows from either $(F)$ or $(J)$, which holds because $\beta^\star$ does not duplicate. (We are a bit imprecise here, since a $\beta$-step might be a $\beta^\star$-step and get erased, but a $B$-step surely can't.) $\odot$

This proves the claim and the theorem. $\odot$

Using the theorem, strong normalisation of $\mathcal{H}$ is reduced to strong normalisation of system F, which is well known ([GLT89]).

This proof allows for easy proofs of some further results in [DCP94].

Adding $\tau$, the terminal type rule, to $\mathcal{H}$ preserves completeness, since the earlier remark still pertains: the terminal type does not have a destructor, therefore $\tau$-steps can always be postponed,

Adding a *left-linear* complete algebraic rewriting system $\mathcal{I}$ (cf. [BTG91]) to $\mathcal{H}$ preserves completeness as well. As in the first-order case, $\mathcal{I}$ is orthogonal to $\mathcal{H}$ so confluence is preserved. By adding the $\mathcal{I}$-rules to $\mathcal{R}$, the proof of the postponement theorem goes through unmodified. We only need to check extra cases where $\mathcal{R}$ appears in the proof, i.e. diagrams $(G)$ and $(J)$. Diagram $(G)$ holds because $\overline{\eta}$ cannot create algebraic redexes (in particular algebraic redexes do not contain $\lambda$'s and are left-linear). Diagram $(J)$ holds because of orthogonality. Completeness can be concluded from completeness without expansion, which is known ([BTG91]).

The system with $E$ is complete because $\overline{\eta}$-postponement still works, hence we can apply Theorem 3.10 of [BTG91].

The system with algebraic rules is complete because $\overline{\eta}$-postponement still works, hence we can apply Theorem 3.10 of [BTG91].

The system with $\overline{\eta}$ is complete by the same method as above.

**Conclusion on Expansion** In coping with the existing results, this section has become somewhat largish. We think this is justified, since we have managed to cover the complete field of expansion as far as we know. Although we have not presented any new results, we have shown our method to proof properties for lambda-calculi with expansions to be quite flexible, and are confident that it applies in other cases as well.

Strong normalisation results are more of a curiosity, because presenting a normalising strategy suffices for most purposes. In all the cases above, first performing reductions and only then expansions is a normalising strategy. This is also a very sensible approach since expansions can be thought of as 'creating form', so make a term bigger (except in the case of the $\overline{\tau}$-rule), hence must be delayed as long as possible. It is only useful in the final 'printout' of a term.

REMARK 6.8. First performing reductions corresponding to the extensional axiomas, then performing the ordinary reductions and only then expansions (corresponding to the extensional rules), would be even better in most cases.

Some problems remain. For example dealing with expansion rules for products in system F (This is a purely theoretical problem, who would want to duplicate an arbitrarily large term, only to throw it away some moments later.) Because of the duplicating behaviour of these rules, the postponement method does not work right away. In general, the problem with system F is that one cannot separate $\beta$ and $B$ in order to prove properties in a modular fashion.

Also expansion rules for higher-order lambda calculi, e.g. the Calculus of Constructions, could be studied. This is especially interesting since confluence for some lambda calculi fails (for good reasons) if the extensional axioms are taken as reduction rules.

## Acknowledgements

We thank Fer-Jan de Vries for asking the question leading to Section 5. We should thank Paul Taylor for his diagrams.

# References

[??94] ??, editor. *ICALP94, Israel*, ??, 1994.

[AGM92] S. Abramsky, Dov M. Gabbay, and T. S. E. Maibaum, editors. *Handbook of Logic in Computer Science*, volume 2, Background: Computational Structures. Oxford University Press, New York, 1992.

[Aka93] Yohji Akama. On Mints' reduction for ccc-calculus. in [BG93, pp. 1–12], 1993.

[Bar84] H. P. Barendregt. *The Lambda Calculus, Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. Elsevier Science Publishers B.V., Amsterdam, revised edition, 1984.

[BG93] M. Bezem and J. F. Groote, editors. *Proceedings of the International Conference on Typed Lambda Calculi and Applications, TLCA'93, March 1993, Utrecht, The Netherlands*, volume 664 of *Lecture Notes in Computer Science*, Berlin Heidelberg, 1993. Springer-Verlag.

[BTG91] V. Breazu-Tannen and J. Gallier. Polymorphic rewriting conserves algebraic strong normalization. *Theoretical Computer Science*, 83(1):3–28, June 1991.

[CF58] Haskell B. Curry and Robert Feys. *Combinatory Logic, volume I.* Studies in Logic and the Foundations of Mathematics. North-Holland, 1958.

[CR36] Alonzo Church and J. B. Rosser. Some properties of conversion. *Transactions of the American Mathematical Society*, 39:472–482, January to June 1936.

[Čub93] Djordje Čubrić. Embedding of a free cartesian closed category into the category of sets. On `triples.math.mcgill.ca` as `pub/cubric/frccc.ps.Z`, April 1993.

[Cur29] H. B. Curry. An analysis of logical substitution. *American Journal of Mathematics*, LI:363–384, 1929.

[Cur30a] H. B. Curry. Grundlagen der kombinatorischen Logik. Teil I. *American Journal of Mathematics*, LII:509–536, 1930.

[Cur30b] H. B. Curry. Grundlagen der kombinatorischen Logik. Teil II. *American Journal of Mathematics*, LII:789–834, 1930.

[DCK93a] Roberto Di Cosmo and Delia Kesner. A confluent reduction for the extensional typed $\lambda$-calculus with pairs, sums, recursion and terminal object. in [LKC93, pp. 645–656], 1993.

[DCK93b] Roberto Di Cosmo and Delia Kesner. Simulating expansions without expansions. Rapport de Recherche 1911, Institut National de Recherche en Informatique et en Automatique, May 1993. To appear in MSCS. Abstract appeared as [DCK93a].

[DCK94] Roberto Di Cosmo and Delia Kesner. Combining first order algebraic rewriting systems, recursion and extensional lambda calculi. in [??94], 1994.

[DCP94] Roberto Di Cosmo and Adolfo Piperno. Expanding extensional polymorphism. On `encore.dsi.uniroma1.it` as `pub/lambda/DiCosmo-Piperno/TLCA95Paper.ps`, September 1994.

[DJ90] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. in [Lee90, Ch. 6 pp. 243–320], 1990.

[Ges90] Alfons Geser. *Relative Termination.* PhD thesis, University of Passau, 1990.

[GLM92] Georges Gonthier, Jean-Jacques Lévy, and Paul-André Melliès. An abstract standardisation theorem. in [LIC92, pp. 72–81], 1992.

[GLT89] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science.* Cambridge University Press, 1989. (1990 reprinting).

[Hei67] J. van Heijenoort, editor. *From Frege to Gödel A Source Book in Mathematical Logic, 1879–1931.* Source Books in the History of the Sciences. Harvard University Press, Cambridge, Massachusetts, 1967.

[Hin67] Roger Hindley. Axioms for strong reduction in combinatory logic. *the Journal of Symbolic Logic*, 32(2):224–236, June 1967.

[Hin94] Roger Hindley, March 1994. Personal communication.

[Klo80] J. W. Klop. *Combinatory Reduction Systems.* PhD thesis, Rijksuniversiteit Utrecht, June 1980. Mathematical Centre Tracts 127.

[Klo92] J. W. Klop. Term rewriting systems. in [AGM92, pp. 1–116], 1992.

[Lee90] Jan van Leeuwen, editor. *Handbook of Theoretical Computer Science*, volume B : Formal Models and Semantics. Elsevier Science Publishers B.V., Amsterdam, 1990.

[LIC92] *Proceedings of the Seventh Annual IEEE Symposium on Logic in Computer Science, Santa Cruz, California, June 22–25, 1992*, Los Alamitos, California, 1992. IEEE Computer Society Press.

[LKC93] A. Lingas, R. Karlsson, and S. Carlsson, editors. *Automata, Languages and Programming, 20th International Colloquium, ICALP93, Lund, Sweden, july 5–9, 1993, Proceedings*, volume 700 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[Min81] G. E. Mints. Closed categories and the theory of proofs. *Journal of Soviet Mathematics*, 15(1):45–62, January 1981. Translation into English of [MO77, pp. 83–114].

[MO77] G. E. Mints and V. P. Orevkov, editors. *Theoretical Applications of the Method of Mathematical Logic, Part II*, volume 68 of *Zapiski Nauchnykh Seminarov Leningradskogo Otdeleniya Matematicheskogo Instituta im. V. A. Steklova AN SSSR*, February 1977.

[NM94] A. Nerode and Yu. V. Matiyasevich, editors. *Logical Foundations of Computer Science, Third International Symposium, LFCS'94, St. Petersburg, Russia, July 1994, Proceedings*, volume 813 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[Oos94] Vincent van Oostrom. *Confluence for Abstract and Higher-Order Rewriting*. PhD thesis, Vrije Universiteit, Amsterdam, March 1994.

[OR94] Vincent van Oostrom and Femke van Raamsdonk. Weak orthogonality implies confluence: the higher-order case. in [NM94, pp. 379–392], 1994.

[Sch24] M. Schönfinkel. Über die Bausteine der mathematischen Logik. *Mathematische Annalen*, 92:305–316, 1924. (English translation in [Hei67, pp. 355-366]).