# On the Power of Simple Diagrams

Roberto Di Cosmo

DMI-LIENS (CNRS URA 1327)
Ecole Normale Supérieure
45, Rue d'Ulm
75230 Paris, France
e-mail:roberto@dicosmo.org

**Abstract.** In this paper we focus on a set of abstract lemmas that are easy to apply and turn out to be quite valuable in order to establish confluence and/or normalization modularly, especially when adding rewriting rules for extensional equalities to various calculi. We show the usefulness of the lemmas by applying them to various systems, ranging from simply typed lambda calculus to higher order lambda calculi, for which we can establish systematically confluence and/or normalization (or decidability of equality) in a simple way. Many result are new, but we also discuss systems for which our technique allows to provide a much simpler proof than what can be found in the literature.

## 1 Introduction

During a recent investigation of confluence and normalization properties of polymorphic lambda calculus with an expansive version of the $\eta$ rule, we came across a nice lemma that gives a simple but quite powerful sufficient condition to check the Church Rosser property for a compound rewriting system in a modular way, providing something of a dual to the usual well-known sufficient condition for the Hindley-Rosen Lemma. Also, under some additional assumptions, it allows to check strong normalization in a modular way. This lemma turns out to allow quite simple and elegant modular proofs of confluence and/or strong normalization for many rewriting systems associated to various lambda calculi.

Our purpose here is to present the lemma and give a survey of applications not only to the case of adding extensional equalities as conditional expansions but also as contractions. More precisely, we will apply it to prove:

- confluence and strong normalization for simple typed lambda calculus with expansive $\eta$ and $SP$, also in the presence of *terminal object* and *iteration*
- confluence and strong normalization for the monadic calculus from [26]
- confluence for the polymorphic lambda calculus with expansive $\eta$ and $SP$, also in the presence of *algebraic term rewriting systems*
- confluence for Girard's $F^\omega$ with expansive $\eta$, even with *algebraic term rewriting systems*
- confluence and strong normalization of the polymorphic extensional typed lambda calculus with Axiom C and contractive $\eta$ from [23]

Of these results, the confluence and normalization for the monadic calculus, confluence for polymorphic lambda calculus with expansions and algebraic TRS's and confluence for Girard's $F^\omega$ with expansions with or without algebraic TRS's are, to the author's best knowledge, entirely new, while for the other results the proofs presented here are quite simpler than the previously published ones known to this author.

## 2 Brief Survey and the Commutation Lemma

First of all, let us recall some basic notation from rewriting theory that we will be using along the exposition.

**Notation 1 (ARS)** *An* abstract reduction system *is a pair* $\langle A, \xrightarrow{R} \rangle$ *of a set* $A$ *and a binary relation* $\xrightarrow{R}$ *on* $A$*. The transitive reflexive closure of a relation* $\xrightarrow{R}$ *is denoted by* $\xrightarrow{R}\!\!\!\!\!\rightarrow$*, while* $\xrightarrow[=]{R}$ *denotes the reflexive closure of* $\xrightarrow{R}$*. When working with different ARS's* $\langle \mathcal{A}, \xrightarrow{R} \rangle$ $\langle A, \xrightarrow{S} \rangle$ *share the same set* $A$*, we will often just talk about reductions* $\xrightarrow{R}$ *and* $\xrightarrow{S}$ *or even* $R$ *and* $S$*. Also,* $R \cup S$ *denotes the reduction obtained as the union of* $R$ *and* $S$*.*

**Definition 2 Commutation of reductions.** Two reductions $\xrightarrow{R}$ and $\xrightarrow{S}$ *commute* with each other if the following diagram holds:

$$\forall a, b, c \in \mathcal{A}, \exists d \in \mathcal{A} \quad \begin{array}{ccc} a & \xrightarrow{R} & c \\ \downarrow{\scriptstyle S} & & \downarrow{\scriptstyle S} \\ b & \xrightarrow[R]{} & d \end{array}$$

**Definition 3 Confluence.** A reduction $\xrightarrow{R}$ is *confluent* if $\xrightarrow{R}\!\!\!\!\!\rightarrow$ commutes with itself, and is *weakly (or locally) confluent* if the following diagram holds:

$$\forall a, b, c \in \mathcal{A}, \exists d \in \mathcal{A} \quad \begin{array}{ccc} a & \xrightarrow{R} & c \\ \downarrow{\scriptstyle S} & & \downarrow{\scriptstyle S} \\ b & \xrightarrow[R]{}\!\!\!\!\!\rightarrow & d \end{array}$$

**Definition 4 Normalization.** For an ARS $\langle A, \xrightarrow{R} \rangle$, we say that $\xrightarrow{R}$ is *strongly normalizing* if, for all $a \in A$, all reduction sequences starting from $a$ are finite. An *$R$-normal form* is an element $a \in A$ such that no reduction out of it is possible. Also, $\xrightarrow{R}$ is *weakly normalizing* if, for all $a \in A$, there is a finite reduction sequence out of $a$ leading to an $R$-normal form.

### 2.1 Modularity of confluence

One of the most known lemmas for showing confluence of rewriting system (especially when they are associated to various lambda calculi) is the following one, due to Hindley and Rosen:

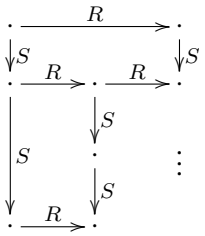
**Lemma 5 Hindley-Rosen ([4], section 3).** *If* $\xrightarrow{R}$ *and* $\xrightarrow{S}$ *are confluent, and* $\xrightarrow{R}\!\!\!\!\!\rightarrow$ *and* $\xrightarrow{S}\!\!\!\!\!\rightarrow$ *commute with each other, then* $R \cup S$ *is confluent.*

Since establishing the commutation directly is often a very difficult task, because one has to cope with arbitrarily long $S$ and $R$ reduction sequences, one does not really use this lemma directly, but via a simpler precondition to commutation:

**Lemma 6 usual sufficient condition for commutation.** *If, whenever* $M \xrightarrow{R} M'$ *and* $M \xrightarrow{S} M''$*, there exist* $M'''$ *s.t.* $M' \xrightarrow{S}\!\!\!\!\!\rightarrow M'''$ *and* $M'' \xrightarrow[=]{R} M'''$*, then* $\xrightarrow{R}\!\!\!\!\!\rightarrow$ *and* $\xrightarrow{S}\!\!\!\!\!\rightarrow$ *commute with each other.*

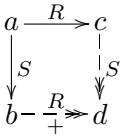
The condition imposed here to use *at most* one step of $R$ reduction to close the diagram is quite restrictive, and is not satisfied for example in the presence of *restricted expansion rules*, that have become quite relevant today for handling extensionality in various lambda calculi (see for example [1, 15, 11, 9, 22]). This restriction is necessary if one does not know anything else about

the two systems $R$ and $S$, as a nice counterexample based on a never ending diagram chase from [4] shows:



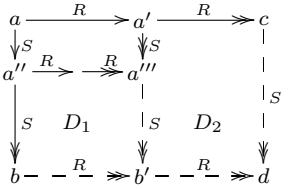
But this very counterexample suggests that, if $R$ is a strongly normalizing system, then we can use a dual condition to the previous one: instead of imposing to use *at most* one step of $R$ reduction ( $\xrightarrow[=]{R}$ ) to close the diagram, one can ask for using *at least* one step of $R$ reduction ( $\xrightarrow[+]{R}$ ). This key observation is at the basis of our original lemma as it was stated in [14], which has a simple, but quite interesting proof (the original lemma assumed implicitly that $R$ is finitely branching, but the version here has not such a restriction, thanks to a very helpful discussion with Pierre Lescanne).

**Lemma 7 Commutation Lemma: a dual sufficient condition for commutation from [14].** *Let $\langle \mathcal{A}, \xrightarrow{R} \rangle$ and $\langle A, \xrightarrow{S} \rangle$ be two abstract reduction systems, where $R$ is strongly normalizing. Let the following diagram hold*

*(DPG)* $\forall a, b, c \in \mathcal{A}, \exists d \in \mathcal{A}$ 

*Then $\xrightarrow{R}$ and $\xrightarrow{S}$ commute.*

*Proof.* Since $R$ is a strongly normalizing rewriting system, we have a well-founded order $<$ on $\mathcal{A}$ by setting $a_1 < a_2$ if $a_2 \xrightarrow{R} a_1$. Also, let us denote $dist(a_1, a_2)$ the length of a given $S$-reduction sequence from $a_1$ to $a_2$. The proof then proceeds by well-founded induction on pairs $(b, dist(a, b))$, ordered lexicographically. Indeed, if $b$ is an $R$-normal form and $dist(a, b) = 0$, then the lemma trivially holds. Otherwise, by hypothesis, there exist $a', a'', a'''$ as in the following diagram.



We can now apply the inductive hypothesis to the diagram $D_1$, since

$$(b, dist(a'', b)) <_{lex} (b, dist(a, b)).$$

Finally, we observe that $b \xrightarrow[+]{R} b'$, just composing the diagram in the hypothesis down from $a$.

Hence we can apply the inductive hypothesis to the diagram $D_2$, since

$$(b', dist(a', b')) <_{lex} (b, dist(a, b)),$$

and we are done.

Alfons Geser remarked this very same property in his PhD Thesis (see [17], page 38, remark after the proof sketch), where the (DPG) diagram is read as $R$ *strictly locally commutes over* $S^{-}1$.

*Remark.* Notice that for (DPG) to hold, it must be the case that the relation $\xleftarrow{S}$ (the inverse of $\xrightarrow{S}$ ) preserve R-normal forms. This is a simple precondition that can be useful to discover that (DPG) does not hold: for example, if $S$ is the usual $\eta$-contraction (see for example [4] for a discussion) and $R$ is simple typed $\beta$, then $xy \xleftarrow{S} (\lambda z : A.xz)y$ and (DPG) does not hold.

### 2.2 Modularity of confluence and/or termination

In[1] Akama gives an interesting lemma to show modularity of both confluence and termination, by requiring some additional conditions on $R$ and $S$, that presents the same difficulty as Hindley-Rosen's lemma, when one tries to use it directly, as the condition on $S$-normal forms requires to handle arbitrarily long $S$-reduction sequences in the hypothesis:

**Lemma 8 [1].** *Let $R$ and $S$ be confluent and strongly normalizing reductions, s.t.*

$$\forall a, b \quad (a \xrightarrow{R} b) \quad \textit{implies} \quad (a^S \xrightarrow[+]{R} b^S),$$

*where $a^S$ and $b^S$ are the $S$-normal forms of $a$ and $b$, respectively; then $R \cup S$ is also confluent and strongly normalizing.*

Here too, we can help improve the situation with a simpler precondition:

**Lemma 9 preconditions for modularity of confluence and/or normalization.**
*Let $\langle \mathcal{A}, \xrightarrow{R} \rangle$ and $\langle \mathcal{A}, \xrightarrow{S} \rangle$ be abstract reduction systems, where $R$-reduction is strongly normalizing. Let the following diagram hold*

$$\forall a, b, c \in \mathcal{A}, \exists d \in \mathcal{A} \quad \begin{array}{ccc} a & \xrightarrow{R} & c \\ \downarrow S & & \vdots\, S \\ b & \xrightarrow[+]{R} & d \end{array}$$

*Then (as seen in 7) $\xrightarrow{R}$ and $\xrightarrow{S}$ commute and furthermore*

- *(i) if $R$ preserves $S$ normal forms (let $S{\downarrow}$ denote reduction to $S$ normal form), then*

$$\forall a, b, c \in \mathcal{A}, \exists d \in \mathcal{A} \quad \begin{array}{ccc} a & \xrightarrow{R} & c \\ \downarrow S{\downarrow} & & \vdots\, S{\downarrow} \\ b & \xrightarrow{R} & d \end{array}$$

- *(ii) if $S$ normal forms are unique and $R$ preserves $S$ normal forms, then*

$$\forall a, b, c, d \in \mathcal{A} \quad \begin{array}{ccc} a & \xrightarrow{R} & c \\ \downarrow S{\downarrow} & & \downarrow S{\downarrow} \\ b & \xrightarrow[+]{R} & d \end{array}$$

*Proof.* The first property can be shown by using 7. As for the second property, notice that by iterating 7 we can obtain:

$$\forall a, b, c \in \mathcal{A}, \exists d \in \mathcal{A} \quad \begin{array}{ccc} a & \xrightarrow{R} & c \\ \downarrow S{\downarrow} & & \vdots\, S{\downarrow} \\ b & \xrightarrow[+]{R} & d' \end{array}$$

Where $d'$ is an $S$-normal form because $R$ preserves $S$-normal forms. But then, by unicity of $S$-normal forms, $d = d'$ and we are done.

The last item tells us that to check the commutation property required by Akama's lemma, which is a global property, as it involves arbitrarily long reduction sequences in the hypothesis, one can resort to just checking the same local condition we had before for commutation, which are usually boring but simple tasks, when $R$ is strongly normalizing, $S$ is confluent (which implies uniqueness of normal forms) *and* $R$ preserves $S$-normal forms (the first two conditions beeing anyway part of the hypothesis of Akama's lemma). This gives

**Corollary 10 Simplified Akama's Lemma.** *Let $S$ and $R$ be confluent and strongly normalizing reductions, s.t.*

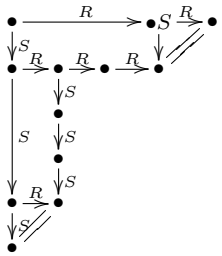
$$\forall a,b,c \in \mathcal{A}, \exists d \in \mathcal{A} \quad \begin{array}{ccc} a & \xrightarrow{R} & c \\ {\scriptstyle S}\downarrow & & \vdots\,{\scriptstyle S} \\ b & \underset{+}{\xrightarrow{R}} & d \end{array}$$

*and $R$ preserves $S$-normal forms: then $S \cup R$ is also confluent and strongly normalizing.*

## 3 Variations of the Lemma that do not work

We can now wonder if it is possible to relax a little the hypothesis of the lemmas, to allow empty $R$ steps at least in some cases. Indeed, in some cases one is interested in combinations of rewriting systems that *both* contain erasers, i.e. rules that can erase redexes in the other system (like for example, contractive $\eta$ together with $\beta$), and this prevents the many-step commutation required as an hypothesis by the lemma.

Consider the following example (the long $=$ signs mean equality of elements):




The strong normalization of $R$ and $S$ do not help here, since the newly built diagram can be exactly identical to the starting one, and one never gets to actually close it.

This is indeed a *counterexample* that rules out a whole bunch of possible even very weak relaxations of the hypothesis in the Lemma, detailed here.

- Allow empty $R$ reduction only when also the closing $S$ reduction is empty, that is reformulating the hypothesis as follows:

$$R \text{ SN}, \quad \forall a,b,c \in \mathcal{A}, \exists d \in \mathcal{A} \quad \begin{array}{ccc} a & \xrightarrow{R} & c \\ {\scriptstyle S}\downarrow & & \vdots\,{\scriptstyle S} \\ b & \underset{+}{\xrightarrow{R}} & d \end{array} \quad \textit{or} \quad \begin{array}{ccc} a & \xrightarrow{R} & b \\ {\scriptstyle S}\downarrow & & \| \\ b & = & b \end{array}$$

  The example above is a counterexample: the only empty $R$ reductions correspond to empty $S$ reductions there.

- One can think that assuming also $S$ strongly normalizing and allowing only simultaneously empty reductions the commutation property may hold. That is, using the following hypothesis:

$$R \text{ SN}, \quad S \text{ SN}, \quad \forall a,b,c \in \mathcal{A}, \exists d \in \mathcal{A} \quad \begin{array}{ccc} a & \xrightarrow{R} & c \\ {\scriptstyle S}\downarrow & & \vdots\,{\scriptstyle S} \\ b & \underset{+}{\xrightarrow{R}} & d \end{array} \quad \textit{or} \quad \begin{array}{ccc} a & \xrightarrow{R} & b \\ {\scriptstyle S}\downarrow & & \| \\ b & = & b \end{array}$$

  Again, the example above respects all the conditions: there are empty $S$ reductions only where empty $R$ reductions appear and vice-versa.

## 4 Applications : simple typed lambda calculus with expansive $\eta$ and $SP$

As a first simple application of the lemma, consider the typed lambda calculus $\lambda^1\beta\eta\pi*$ for Cartesian Closed Categories: this consists of $\beta, \eta, \pi, SP$ and a rule $Top$ that collapses all terms of a special type $T$ into a single constant $*$ (with both $\eta$ and $SP$ taken as expansions). A discussion of the conditional expansion rules falls outside the scope of this work (the interested reader will find a thorough discussion and motivation for example in [11]), but let us just point out that using the traditional contractive rules for $\eta$ and $SP$, the system as it is not even confluent, and one has to go through a lot of hassle to complete it to a confluent one [10]. It is worth noting that the same problem for confluence comes up with algebraic rewriting rules for constant functions like $f(x) \longrightarrow a$.

$$
\begin{array}{lll}
(\beta) & (\lambda x : A.M)N \xrightarrow{\beta} M[N/x] & \\
(\pi_i) & \pi_i\langle M_1, M_2\rangle \xrightarrow{\pi_i} M_i, & \text{for } i = 1, 2 \\
(SP) & M \xrightarrow{SP} \langle \pi_1(M), \pi_2(M)\rangle, & \text{if } \begin{cases} M : A \times B \\ M \text{ is not a pair and is not projected} \end{cases} \\
(\eta) & M \xrightarrow{\eta} \lambda x : A.Mx, & \text{if } \begin{cases} x \ fresh \\ M : A \Rightarrow C \\ M \text{ is not a } \lambda\text{-abstraction} \\ M \text{ is not applied} \end{cases} \\
(Top) & M \xrightarrow{Top} *, & \text{if } M : T \text{ and } M \not\equiv *
\end{array}
$$

**Table 1.** Reduction system for simple typed lambda calculus with expansions and terminal object.

There have been many different proofs of confluence and strong normalization in the literature for this calculus (or some variations of it) (for example [1, 15, 11, 9, 22]), but all of them are essentially technically complex exercises, with only [1, 11] using some kind of modular technique, yet requiring a serious amount of work.

Here our lemma suggests the following proof.

**Theorem 11.** *Simple typed $\lambda$-calculus with expansions and terminal object is confluent and strongly normalizing.*

*Proof.* It is easy to verify that rules $\eta$ and SP do not erase any redex, while the rules $\beta$ and $\pi$ and $Top$ preserve the normal forms of $\eta$ and SP. Then it is quite natural to set $R = \beta \cup \pi \cup Top$ and $S = \eta \cup SP$, and try to apply our lemma 9. This boils down to checking a small subset of the diagrams one should check for the local confluence of the whole system (which is not a very easy task, because the reduction is no longer a congruence, but is unavoidable in any other proof technique[1]).

Then, since confluence and strong normalization for the two separate subsystems are already well known (and easy to show with traditional techniques), we can finally apply Akama's lemma and get confluence and normalization for the full system.

This gives an extremely simple and straightforward proof which is way easier than the already published ones.

[1] Actually, one can simply go over the relevant cases in [11], where local confluence is checked in detail, and verify that the *at least one step* condition is indeed respected.

### 4.1 Handling Iteration

As was originally remarked in [1], one is faced with serious technical difficulties when trying to use *directly* Akama's lemma to handle a weaker computational principle, namely iteration:

$$It(a, f, 0) \longrightarrow a \qquad It(a, f, S(e)) \longrightarrow f(It(a, f, e))$$

Indeed, one gets involved in a complex technical analysis of the shape of expansive normal forms that does not behave well when we add iteration.

Nevertheless, here again our simple precondition applies with no difficulty, and one gets confluence and strong normalization (local confluence is easy to check even with expansions, as the only nontrivial divergence, namely an expansion of $f$, can be closed by using $\beta$).

It is worth recalling here that using a modular technique presented in [12], it is now quite easy to show that the previous systems stays confluent if we add a recursion operator.

## 5 The monadic calculus for database query languages

This calculus, that arises from category theoretic considerations and forms the basis for an elegant database query language, was first introduced in [26]. An equivalent calculus NRC (see table 3)*without* these two last features has been proven confluent and strongly normalizing by Woong in his PhD thesis [28]. It contains a subset of the simple typed lambda calculus we have seen above, as it provides a limited form of $\beta$ reduction (arguments of functions cannot be functions themselves), an equality axiom for the terminal object and the extensional equality axiom for pairs (SP) and functions ($\eta$).

$$\frac{}{\{\} : \{s\}} \qquad \frac{e : s}{\{e\} : \{s\}} \qquad \frac{\{e_1\} : \{s\} \quad \{e_2\} : \{s\}}{\{e_1 \cup e_2\} : \{s\}} \qquad \frac{\{e_1\} : \{s\} \quad \{e_2\} : \{t\}}{\bigcup \{e_1 | x \in e_2\} : \{s\}}$$

**Table 2.** The typing rules for sets in NRC

But it also provides constructors and operations to manipulate *sets* of values (terms and types for sets ($\{\}$), union ($\cup$) and a form of set comprehension ($\bigcup \{e_1 | x \in e_2\}$).
We are now able to state our result:

**Theorem 12.** *The reduction system for the monadic calulus with expansive $\eta$ and $SP$ is confluent and strongly normalizing.*

*Proof.* Take $R$ to be the system proved CR and SN by Woong (that is the system of table 3 without expansions), and $S$ to be expansive $SP$ and $\eta$ rules alone. The (DPG) diagram is easily checked, as expansive $SP$ and $\eta$ does not erase any $R$ redex. Since $R$ is SN and $SP \cup \eta$ is known to be CR and SN, this is enough to get confluence for the system with expansive $SP$ using 7. It is very easy to check that $R$ preserves $SP \cup \eta$ expansive normal forms: all rules in $R$ preserve types (this ensure that no new redex due to types is created) and no rule can move a subterm from a position where an expansion is not legal to one where it is legal (the substitution rules can destroy expansion redexes, but not create them). So we get also strong normalization for the full system, using 9.

$$
\begin{array}{lll}
(\beta) & (\lambda x : A.M)N \xrightarrow{\beta} M[N/x] & (A \text{ not a functional type}) \\
(\pi_i) & \pi_i\langle M_1, M_2\rangle \xrightarrow{\pi_i} M_i, & \text{for } i = 1, 2 \\
(SP) & M \xrightarrow{SP} \langle \pi_1(M), \pi_2(M)\rangle, & \text{if } \begin{cases} M : A \times B \\ M \text{ is not a pair and is not projected} \end{cases} \\
(\eta) & M \xrightarrow{\eta} \lambda x : A.Mx, & \text{if } \begin{cases} x\ fresh \\ M : A \Rightarrow C \\ M \text{ is not a } \lambda\text{-abstraction} \\ M \text{ is not applied} \end{cases} \\
(Top) & M \xrightarrow{Top} *, & \text{if } M : T \text{ and } M \not\equiv *
\end{array}
$$

$$(Set\ monad\ operations)$$

$$
\begin{array}{ll}
(empty) & \bigcup\{e|x \in \emptyset\} \longrightarrow \emptyset \\
(flat) & \bigcup\{e_1|x \in \{e_2\}\} \longrightarrow e_1[e_2/x] \\
(distrib) & \bigcup\{e|x \in (e_1 \cup e_2)\} \longrightarrow \bigcup\{e|x \in e_1\} \cup \bigcup\{e|x \in e_2\} \\
(assoc) & \bigcup\{e_1|x \in \bigcup\{e_2|y \in e_3\}\} \longrightarrow \bigcup\{\bigcup\{e_1|x \in e_2\}|y \in e_3\}
\end{array}
$$

**Table 3.** The reduction system for the monadic query calculus NRC

## 6 The polymorphic lambda calculus with expansive $SP$, $\eta$ and $\eta^2$

The polymorphic lambda calculus (also known as Girard's System $F$, see [20]) adds to the simple typed lambda calculus the possibility of taking types as parameters, via type abstraction $\Lambda X.M$ and type application $M[A]$. The essential features from the rewriting point of view are a new $\beta^2$ rule that is analogous to $\beta$, but operates on types, and a *contractive* extensional rule $\eta_c^2$:

$$
\begin{array}{ll}
(\beta^2) & (\Lambda X.M)[A] \xrightarrow{\beta^2} M[A/X] \\
(\eta_c^2) & (\Lambda X.M[X]) \xrightarrow{\eta_c^2} M \quad (\text{if } X \notin FTV(M))
\end{array}
$$

where $FTV(M)$ is the set of free type variables of $M$.
For the same reasons why expansions are recognized as a necessity for first order calculi, one would also like better to use an *expansive* rule for $\eta^2$

$$
(\eta^2)\ \ M \xrightarrow{\eta^2} (\Lambda X.M[X]) \text{ if } \begin{cases} X fresh \\ M : \forall X.A \\ M \text{ is not a polymorphic } \lambda\text{-abstraction} \\ M \text{ is not applied} \end{cases}
$$

Now, our simple lemmas allow us to derive in a very straightforward way the confluence of this system with expansion rules.

**Theorem 13 Confluence with expansions.** *The polymorphic lambda calculus with expansive $SP$, $\eta$ and $\eta^2$ is confluent.*

*Proof.* First of all, notice that $SP \cup \eta \cup \eta^2$ is confluent, as it enjoys the diamond property. Now, for the full calculus, take $R$ to be the usual polymorphic lambda calculus without expansion rules, which we know is confluent and strongly normalizing, and let $S$ be the system made up of the expansion rules alone ($SP$, $\eta$ and $\eta^2$). It is an easy task to check (DPG), as the only new cases are due to $\eta^2$ and $\beta^2$ (see [13]), and the expansion rules do not erase $R$ redexes. Again, we can apply 7, and confluence for the full system follows.

It should be noted that the strategy consisting in doing all non-expansive steps first and then only expansions is normalizing, so this very simple proof (that gives us confluence) is already enough both for getting decidability of equality and getting the unicity of polymorphic $\beta\eta$-long normal forms, which is useful in higher order unification [21].

### 6.1 Handling confluent algebraic term rewriting systems (TRS's)

It is also possible to go on further and show that whenever we have a canonical (that is, confluent and strongly normalizing) algebraic TRS, then it can be added to system F with expansion rules, preserving decidability of equality. One important property we will use is the following, that holds for arbitrary TRS's:

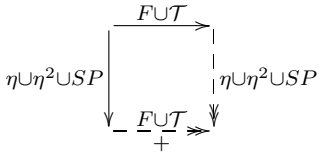
**Lemma 14 Algebraic reduction commutes with reduction to expansive normal form.** *Let $M$, $M'$ be arbitrary terms, and $M^E$, $M'^E$ be their repective expansive normal forms. Then whenever $M \xrightarrow{\mathcal{T}} M'$, we have $M^E \xrightarrow{\mathcal{T}} M'^E$.*

*Proof.* A simple induction on the structure of terms, using in the crucial case the fact (proven by induction on the structure of algebraic terms) that $(A[M/x])^E = A[M^E/x]$ for any *algebraic* term $A$.

Now, we will first show a simple and self-contained proof technique that works only in the case that the rewriting system is also *left-linear* (i.e. when variables occur at most once in the l.h.s. of any algebraic rewriting rule):

**Theorem 15 Expansive System F plus left-linear TRS's.** *Let $\mathcal{T}$ be a left-linear algebraic TRS which is confluent and strongly normalizing. Then System F with expansions together with $\mathcal{T}$ forms a confluent system.*

*Proof.* We already established that (DPG) holds taking system $F$ as the horizontal system and expansions as the vertical one. Left linearity makes it easy to show that (DPG) holds also taking the algebraic system $\mathcal{T}$ as the horizontal reduction and expansions as the vertical reduction. Taken together, these two facts give




We know from [6, 8] that combining the non-extensional simply typed lambda calculus with a confluent first-order algebraic rewriting system preserves confluence. On the other hand, this combination yields a strongly normalizing system when the algebraic one is [7, 25]. This is enough to apply lemma 7 and obtain confluence of $F$ with expansions together with $\mathcal{T}$.

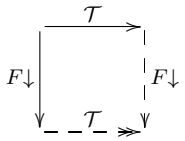
**Corollary 16.** *System F together with a left-linear canonical TRS is a decidable system.*

*Proof.* The expansions preserve also algebraic normal forms (because the system is left linear), and the strategy consisting in going to $F \cup \mathcal{T}$ normal form first and then normalize w.r.t. the expansion rules is normalizing. This, together with confluence, gives a decision procedure for equality.

#### Handling non left-linear TRS's

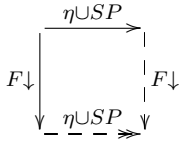
The restriction to left-linear TRS's is imposed here by the necessity to ensure that (DPG) holds, which cannot be the case in the presence of non-left-linear rules: a vertical reduction could destroy an horizontal redex. But it is possible to raise this restriction by using some technical results from [8]:

there it is shown that algebraic reductions commute with reduction to normal form in $F$ without extensional rules(which we write here $F\downarrow$)

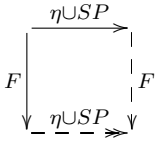



We can show the same result w.r.t. expansion rules.

**Lemma 17 Expansions commute with reduction to $F$ normal form.** *Reduction to $F$ normal form commutes w.r.t. expansion rules, i.e. the following diagram holds:*




*Proof.* We have shown above, by establishing (DPG) and using our commutation lemma, that expansions commute with the reductions in $F$ without extensional rules, that is

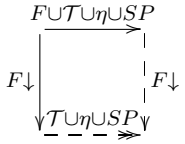



Now the result is a direct consequence of the fact that expansions preserve $F$-normal forms (the restriction are there exactly to insure this).

We can now state the main result:

**Theorem 18 Expansive System $F$ with confluent TRS's.** *System $F$ plus expansion rules plus an* arbitrary *confluent TRS $\mathcal{T}$ is confluent.*

*Proof.* Lemmas 17 together with 14 and lemma 4.1 of [8] (which states that algebraic reduction commutes with reduction to $\beta$ normal form) allow us to establish the following simulation property:




since $F\downarrow$ is confluent, this allows to reduce the confluence of $F \cup \mathcal{T} \cup \eta \cup SP$ to confluence of $\mathcal{T} \cup \eta \cup SP$, which can be in turn reduced, due to the confluence of expansion rules, to confluence of $\mathcal{T}$ via the simulation established in 14. But $\mathcal{T}$ is confluent by hypothesis, and we are done.

It is worth noting that the normal forms in this rewriting system are exactly Huet's second order *long $\beta\eta$ normal forms*.

## 7 Girard's $F^\omega$ with expansion rules

A quite surprising fact, the proof strategy we used to show decidability of $F$ plus expansion rules even in the presence of canonical left linear TRS's can be used with no changes at all to show decidability of Girard's $F^\omega$ with expansive $\eta$, even with left linear canonical TRS's added. We do not fully introduce here the syntax and typing judgements for System $F^\omega$ (see [16] for a detailed introduction to the topic), but let's recall that this system is basically System $F$ with a simple typed lambda calculus *over*

*its types*, the types of the types being now called *kinds*. More formally, kinds, types and terms are defined by the following grammar:

$$\begin{aligned}
(Kinds)\ \ & K := *|K \to K \\
(Types)\ \ & T := t|A|T \Rightarrow T|\forall t : K.T|\lambda t : K.T|T\,T \\
(Terms)\ \ & M := x|\lambda x : T.M|M\,M|\Lambda t : K.M|M[T]
\end{aligned}$$

and one only works with those types that kind-check and terms that type-check w.r.t. appropriate kinding and typing rules (here we follow essentially the presentation from [16]).

$$\frac{\Gamma, t : K_1 \vdash s : K_2}{\Gamma \vdash (\lambda t : K_1.s) : K_1 \to K_2} \qquad \frac{\Gamma \vdash t : K_1 \to K_2 \quad \Gamma \vdash s : K_1}{\Gamma \vdash ts : K_2}$$

$$\frac{\Gamma, t : * \vdash s : *}{\Gamma \vdash \forall t : *.s : *} \qquad \frac{\Gamma \vdash t : * \quad \Gamma \vdash s : *}{\Gamma \vdash t \Rightarrow s : *}$$

**Table 4.** Kinding judgements

Over the types, that now form a simple typed lambda calculus, we have the usual $\beta$ and $\eta$ equality, that we turn into rewriting by choosing the usual $\beta$-reduction and restricted expansion rule for $\eta$. Once the well-kinded types are defined, one defines the well-typed terms as in table 5.

$$\frac{\Gamma, x : t_1 \vdash M : t_2}{\Gamma \vdash (\lambda x : t_1.M) : t_1 \Rightarrow t_2} \qquad \frac{\Gamma \vdash M : t_1 \Rightarrow t_2 \quad \Gamma \vdash N : t_1}{\Gamma \vdash MN : t_2}$$

$$\frac{\Gamma, t_1 : K \vdash M : t_2}{\Gamma \vdash \Lambda t_1 : K.M : \forall t_1 : K.t_2} \qquad \frac{\Gamma \vdash M : \forall t_1 : K.t_2 \quad \Gamma \vdash s : K}{\Gamma \vdash M[s] : t_2[s/t_1]}$$

$$\frac{\Gamma \vdash M : t \quad t =_{\beta\eta} s}{\Gamma \vdash M : s}$$

**Table 5.** Typing judgements

Over terms, one has the usual $\beta$ reduction, both for term application and for type application. The most remarkable fact is that now a term has no longer a *unique* type, and this is a fact that we need to consider when defining expansion rules.
We have no difficulty in writing the higher order $\eta$-expansion rule by simply generalizing the one for System F:

$$(\eta^\omega)\ \ M \xrightarrow{\eta^\omega} (\Lambda t : K.M[t]) \text{ if } \begin{cases} t\ fresh \\ M : (\forall t : K.A) \\ M \text{ is not a polymorphic } \lambda\text{-abstraction} \\ M \text{ is not applied} \end{cases}$$

But for the first order expansion, due to the type conversion rules, the usual $\eta$ expansion rule taken alone is now not even confluent, as it can be the case that:

$$\lambda x : A'.Mx \xleftarrow{\eta} M \xrightarrow{\eta} \lambda x : A.Mx$$

where we only know that $A =_{\beta\eta} A'$ in the type-conversion relation.
For this reason, we chose to work with a somewhat more restrictive rule, that only allow expansion with types in normal form w.r.t. the simple typed lambda calculus over types.

$$(\overline{\eta})\ M \xrightarrow{\eta} \lambda x : A.Mx,\ \text{if}\ \begin{cases} x\ fresh \\ M : A{\Rightarrow}C, \textit{with } A{\Rightarrow}C \textit{ in type normal form} \\ M \text{ is not a } \lambda\text{-abstraction} \\ M \text{ is not applied} \end{cases}$$

Let's call $F^{\omega}_{exp}$ the rewriting system composed by the usual rules for $F^{\omega}$ plus expansive $\eta$ and $\eta^{\omega}$, and call $F^{\omega}_{\overline{exp}}$ the system $F^{\omega}_{exp}$ with our limited expansion rule $\overline{\eta}$ instead of $\eta$. The choice of a limited version of $\eta$ expansion does not make us loose any equality.

**Lemma 19 $F^{\omega}_{\overline{exp}}$ and $F^{\omega}_{exp}$ vs. $F^{\omega}$-equality.** *The reflexive, symmetric and transitive closure of $\xrightarrow{F^{\omega}_{exp}}$ generates the usual equality over terms of $F^{\omega}$. The same holds for $\xrightarrow{F^{\omega}_{\overline{exp}}}$.*

*Proof.* This comes from the fact that all $\eta$ equalities $M = \lambda x : A.Mx$ that seem to be forbidden by our restrictions on the expansions can be obtained either by $\beta$-reduction of $\lambda x : A.Mx$ ( both for $F^{\omega}_{exp}$ and $F^{\omega}_{\overline{exp}}$) or by the *type reduction* (which we know is confluent) of $A$ (needed for $F^{\omega}_{\overline{exp}}$).

Now, for this system, we can use the same proof strategy as for System $F$:

**Theorem 20 Confluence with expansions.** *System $F^{\omega}_{\overline{exp}}$ is confluent and weakly normalizing (thus decidable).*

*Proof.* The proof proceeds exactly as for System $F$ (the only novelty is the need to show that $\eta \cup \eta^{\omega}$ is confluent, which is trivial as they enjoy the diamond property).

Much in the same way as for System $F$, we can then also establish the following:

**Theorem 21 System $F^{\omega}_{\overline{exp}}$ plus left-linear TRS's.** *Let T be a left-linear algebraic TRS which is confluent and strongly normalizing. Then $F^{\omega}_{\overline{exp}} \cup \mathcal{T}$ forms a confluent and weakly normalizing (thus decidable) system.*

Indeed, we can now even prove the following:

**Corollary 22 Confluence with general $\eta$-expansion.** *The system $F^{\omega}_{exp}$ (where $\eta$ is not restricted to type normal forms) is confluent and weakly normalizing.*

*Proof.* Consider a divergence $M' \xleftarrow{} M \xrightarrow{} M''$ in the system $F^{\omega}_{exp}$. This means that $M' = M''$, and since the equality generated by $F^{\omega}_{exp}$ is the same as the usual one for $F^{\omega}$, we have that $M' \xrightarrow{F^{\omega}_{exp}} M''' \xleftarrow{F^{\omega}_{exp}} M''$, and since an expansion on type-normal form is a special case of the non-restricted one, this is also $M' \xrightarrow{F^{\omega}_{exp}} M''' \xleftarrow{F^{\omega}_{exp}} M''$.
As for normalization, the same strategy as for $F^{\omega}_{\overline{exp}}$ will obviously do.

**Corollary 23.** *The union of the system $F^{\omega}_{exp}$ with a canonical left-linear TRS is confluent and weakly normalizing (hence decidable).*

## 8 The polymorphic lambda calculus with Axiom C

This calculus stems from a promising new analysis of parametricity proposed in [23], where it is shown that it is sound to add the following axiom C to the polymorphic lambda calculus (system F):

$$(Axiom\ C) \qquad \frac{\Gamma| - M : \forall X.\alpha, \quad X \notin FV(\Gamma) \cup FV(\alpha)}{M\sigma = M\tau}$$

Where $FV(M)$ is the set of free variables of $M$. It has been long open the problem to prove that the equational theory of the resulting system $F_C$ is decidable, which can be done for example showing that the usual reduction rules for system F plus the following new ones form a confluent and normalizing system:

$$(\beta^2_C)\ M[\sigma/X] \longrightarrow M[\forall X.X/X]\ (M:\alpha, X \notin FTV(\alpha), \sigma \neq \forall X.X)$$
$$(\eta^2_C) \quad \Lambda X.M[\forall X.X] \longrightarrow M \quad (M:\forall Y.\alpha, Y \notin FV(\alpha), X \notin FV(M))$$

Only recently in [5] it has been proved that this system is indeed CR and SN, using a non modular approach. We show here how, using our simple lemma, we can get the decidability of equality in $F_C$ in a very straightforward manner (via confluence and weak normalization). Let us start with the system $F_C$ without the extensional rules $\eta^2$ and $\eta^2_C$. We apply our technique taking system $F$ without $\eta^2$ (we will denote it $F'$) as $R$ (the horizontal reduction) and just $\beta^2_C$ as $S$ (the vertical reduction).

**Lemma 24.** *System $F'$ plus $\beta^2_C$ is confluent and strongly normalizing.*

*Proof.* The two systems are separately confluent and strongly normalizing (normalization for $\beta^2_C$ is trivial as each reduction strictly decreases the number of redexes, while confluence comes from a rather sophisticated result in [23], but again this is out of the scope of the present paper). The commutation can be easily checked, and the *at least one step* is guaranteed by the fact that $\beta^2_C$ does not erase redexes of $F'$, as no reduction in this system depends on the particular form of a type (which is not the case of $\eta^2$). Finally, it is easily seen that system $F'$ preserves $\beta^2_C$ normal forms. Then, we can apply our lemma 9 and Akama's lemma and we are done.

Then we focus on a restricted version of the rules $\eta^2_C$ and $\eta^2$:

$$(\eta^{2\prime}_C)\ \Lambda X.M[\forall X.X] \longrightarrow M \text{ if } \begin{cases} X \notin FTV(M) \\ M \neq \Lambda Z.M' \text{ with } Z \notin FTV(M') \\ \Lambda X.M[\forall X.X] \text{ is not applied to the type } \forall X.X \end{cases}$$

$$(\eta^{2\prime}) \quad \Lambda X.M[X] \longrightarrow M \quad \text{if } \begin{cases} X \notin FTV(M) \\ M \neq \Lambda Z.M' \\ \Lambda X.M[X] \text{ is not applied to a type} \end{cases}$$

**Lemma 25.** *The system $\eta^{2\prime} \cup \eta^{2\prime}_C$ is strongly normalizing and confluent.*

*Proof.* Strong normalization is trivial, as the rules decrease the number of $\Lambda$'s in a term. Confluence is also easy, as the system has the diamond property.

We are now in a position to state the main results:

**Theorem 26.** *The rewriting system for $F_C$ is confluent.*

*Proof.* For confluence, take $R$ as system $F' \cup \beta^2_C$ and $S$ as $\eta^{2\prime} \cup \eta^{2\prime}_C$: it is easy to check the (DPG) diagram, where the at least one step is guaranteed by the restrictions imposed on $\eta^{2\prime}$ and $\eta^{2\prime}_C$, and then we get confluence of the system $F' \cup \beta^2_C \cup \eta^{2\prime} \cup \eta^{2\prime}_C$ using lemma 7. But it is quite easy to check that if $M \xrightarrow{\eta^2_C} M'$, then we have either $M \xrightarrow{\eta^{2\prime}_C} M'$ or $M \xrightarrow{\beta^2} M'$, and that if $M \xrightarrow{\eta^2} M'$, then either $M \xrightarrow{\eta^{2\prime}} M'$or $M \xrightarrow{\beta^2} M'$, so $F' \cup \beta^2_C \cup \eta^{2\prime} \cup \eta^{2\prime}_C$ is the same as $F \cup \beta^2_C \cup \eta^2_C$ and we are done.

To show that equality in $F_C$ is decidable, it is enough to provide a normalizing strategy (like the one that does $\eta^2 \cup \beta^2_C \cup \eta^2_C$ after $F'$), but we are able to show more:

**Theorem 27.** *The rewriting system for $F_C$ is strongly normalizing.*

*Proof.* Since $\beta^2$ does not preserve $\eta^2_C$ normal forms, we cannot obtain strong normalization *directly* from our lemma, but the commutation we have shown using our lemma between $\beta^2_C$ and $F'$ allow to obtain the result indirectly via a sort of postponement of $\eta^2 \cup \eta^2_C$. Indeed, in the system $F_C$ $\eta^2$ can be postponed to any other rule, while it is possible to show that from any infinite reduction containing $\eta^2_C$ one can build an infinite reduction not containing it. The only case when $\eta^2_C$ cannot be simply postponed arises when we have a reduction sequence

$$C[(\Lambda X.(\Lambda Y.M)[\forall X.X])[A]] \xrightarrow{\eta^2_C} C[(\Lambda Y.M)[A]] \xrightarrow{\beta^2} C[M[A/Y]]$$

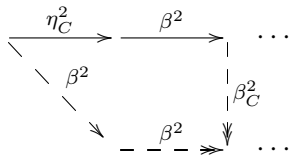
and then the only thing we can do to perform $\beta^2$ first is either

$$C[(\Lambda X.(\Lambda Y.M)[\forall X.X])[A]] \xrightarrow{\beta^2} C[(\Lambda X.M[\forall X.X/Y])[A]] \xrightarrow{\beta^2} C[M[\forall X.X/Y]]$$

where the last step uses the fact that $X \notin FTV(M)$, or

$$C[(\Lambda X.(\Lambda Y.M)[\forall X.X])[A]] \xrightarrow{\beta^2} C[(\Lambda Y.M)[\forall X.X]] \xrightarrow{\beta^2} C[M[\forall X.X/Y]]$$

where the first step uses the fact that $X \notin FTV(M)$. In any case, we did not achieve a real postponement, as we do not get to $C[M[A/Y]]$. Nevertheless, remark that an infinite reduction sequence can be projected via $\beta^2_C$ into another infinite reduction sequence (using (DPG) as established in 24 this is quite easy), and then we can proceed as follows to build an infinite sequence without $\eta^2_C$ from an infinite sequence containing it: postpone the rule whenever possible, and when it is not possible we can build the diagram




where we have managed to bring to front at least one $\beta^2$ step, while still having an infinite sequence available (the one projected via $\beta^2_C$ is still infinite). This is enough to reduce normalization of $F_C$ to the already known normalization for $F$.

## 9 Conclusions

We have studied a few lemmas for proving the commutation of two rewriting relations and/or the preservation of strong normalization. Despite their extreme simplicity, we showed that they can be of great utility in proving confluence and/or normalization of many rewriting systems associated to various typed lambda calculi, especially (but not only) when one needs to use expansive rewriting rules for $\eta$ and surjective pairing in order to get a confluent system in the presence of rules like $Top$ or of general algebraic rewriting systems. The major advantages of the lemmas are the simplicity of the preconditions that one needs to establish. This has allowed us to collect in just one paper a survey of results that, with the traditional approaches, would have required (or have required, for the old results like the simple typed lambda calculus), a full paper by themselves. It is worth mentioning that what we presented here is also a relevant contribution to the study of expansion rules in rewriting with typed lambda calculi, which are now widely used.

*Acknowledgements* I am endebted to Delia Kesner, for many discussions and comments on the whole work, and to Adolfo Piperno for many pleasurable discussions on the rewriting lemma.

## References


1. Y. Akama. On Mints' reductions for ccc-Calculus. In *TLCA*, n. 664 in LNCS, pages 1–12. Springer Verlag, 1993.
2. F. Barbanera. Combining term-rewriting and type-assignment systems. In *3rd It. Conf. on TCS*, 1989.
3. F. Barbanera, M. Fernandez, and H. Geuvers. Modularity of strong normalization and confluence in the algebraic-$\lambda$-cube. In *LICS*, Paris, 1994.
4. H. Barendregt. *The Lambda Calculus; Its syntax and Semantics (revised edition)*. North Holland, 1984.
5. G. Bellè. Syntactical properties of an extension of girard's system $f$ where types can be taken as "generic" inputs. 1995. Available as `ftp://idefix.disi.unige.it/pub/gbelle/systemFC.ps.Z`.
6. V. Breazu-Tannen. Combining algebra and higher order types. In *LICS*, pages 82–90, July 1988.
7. V. Breazu-Tannen and J. Gallier. Polymorphic rewriting preserves algebraic strong normalization. *TCS*, 83:3–28, 1991.
8. V. Breazu-Tannen and J. Gallier. Polymorphic rewiting preserves algebraic confluence. *Inf. and Comp.*, 114:1–29, 1994.
9. D. Cubric. On free CCC. Distributed on the `types` mailing list, 1992.
10. P.-L. Curien and R. Di Cosmo. A confluent reduction system for the $\lambda$-calculus with surjective pairing and terminal object. *JFP*, 1995. To appear. A preliminary version appeared in *ICALP* 91.
11. R. Di Cosmo and D. Kesner. Simulating expansions without expansions. *MSCS*, 4:1–48, 1994.
12. R. Di Cosmo and D. Kesner. Combining algebraic rewriting, extensional lambda calculi and fixpoints. *TCS*, 1995. To appear.
13. R. Di Cosmo and D. Kesner. Rewriting with polymorphic extensional $\lambda$-calculus. In *CSL'95 (extended abstract)*, 1995. Full version accepted for CSL95 Proceedings, to appear in 1996.
14. R. Di Cosmo and A. Piperno. Expanding extensional polymorphism. In M. Dezani-Ciancaglini and G. Plotkin, editors, *TLCA*, volume 902 of *LNCS*, pages 139–153, Apr. 1995.
15. D. J. Dougherty. Some lambda calculi with categorical sums and products. In *RTA*, 1993.
16. J. Gallier. *On Girard's "Candidats de Reductibilité"*, pages 123–203. Logic and Computer Science. Academic Press, 1990.
17. A. Geser. *Relative termination*. Dissertation, Fakultät für Mathematik und Informatik, Universität Passau, Germany, 1990.
18. N. Ghani. $\beta\eta$-equality for coproducts. In M. Dezani-Ciancaglini and G. Plotkin, editors, *TLCA*, volume 902 of *LNCS*, 1995.
19. N. Ghani. Extensionality and polymorphism. University of Edimburgh, Submitted, 1995.
20. J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge University Press, 1990.
21. G. Huet. Résolution d'équations dans les langages d'ordre $1, 2, \ldots, \omega$. *Thèse d'Etat, Université Paris VII*, 1976.
22. C. B. Jay and N. Ghani. The Virtues of Eta-expansion. *JFP*, 5(2):135–154, Apr. 1995.
23. G. Longo, K. Milsted, and S. Soloviev. The Genericity Theorem and effective Parametricity in Polymorphic lambda-calculus. *TCS*, 121:323–349, 1993.
24. G. Mints. Teorija categorii i teoria dokazatelstv.I. *Aktualnye problemy logiki i metodologii nauky*, pages 252–278, 1979.
25. M. Okada. Strong normalizability for the combined system of the types lambda calculus and an arbitrary convergent term rewrite system. In *Symp. Symb. and Alg. Comp.*, 1989.
26. V. Tannen, P. Buneman, and L. Wong. Naturally embedded query languages. In *4th Int. Conf. on Database Theory*, n. 646 in LNCS, 1992. Available as `ftp://www.cis.upenn.edu/pub/papers/db-research/icdt92.dvi.Z`.
27. V. van Oostrom. Developing developments. Draft, 1994.
28. L. Wong. *Querying nested collections*. PhD thesis, University of Pennsylvania, 1994. Available as `ftp://www.cis.upenn.edu/pub/papers/db-research/limsoonphd.ps.Z`.


This article was processed using the LaTeX macro package with LLNCS style