A confluent reduction for the extensional typed $\lambda-$calculus with pairs, sums, recursion and terminal object

Roberto Di Cosmo* Delia Kesner §

Abstract

We add extensional equalities for the functional and product types to the typed λ-calculus with not only products and terminal object, but also sums and bounded recursion (a version of recursion that does not allow recursive calls of infinite length). We provide a confluent and strongly normalizing (thus decidable) rewriting system for the calculus, that stays confluent when allowing unbounded recursion. For that, we turn the extensional equalities into *expansion* rules, and not into contractions as is done traditionally. We first prove the calculus to be weakly confluent, which is a more complex and interesting task than for the usual λ-calculus. Then we provide an effective mechanism to simulate expansions without expansion rules, so that the strong normalization of the calculus can be derived from that of the underlying, traditional, non extensional system. These results give us the confluence of the full calculus, but we also show how to deduce confluence directly form our simulation technique, without the weak confluence property.

1 Introduction

Over the past years there has been a growing interest in the properties of λ-calculus extended with various different type constructors, in particular pairs and sums, used to represent common data types. For these type constructors it is customary to provide a set of equalities that are then turned into computation rules: this is the case, for example, of the elimination rules for pairs:

$$(\pi_1) \quad \pi_1(\langle M, N\rangle) \quad = \quad M \qquad (\pi_2) \quad \pi_2(\langle M, N\rangle) \quad = \quad N$$

They tell us how to operationally *compute* with objects of these types: if we have a pair $\langle M, N\rangle$, then we can decompose it to access its first or second component.

There is anyway something else that one likes to do with λ-calculus, besides using λ-terms as programs to be computed: one would like to *reason* about programs, to prove that they enjoy certain properties. Here is where extensional equalities come into play. In the case of functions, for example, since the only operational way to *use* a function is to apply it to an argument, we do not really want to consider a term M of function type different from the term $\lambda x.Mx$ where x does not occur free in M: both terms, when applied to an argument N, give the same result MN. Similarly for pairs, the only operational way to *use* a pair is by projecting out the first or the second component, so we do not want to consider a term M of product type different from the term $\langle \pi_1(M), \pi_2(M)\rangle$: the result of accessing any of these two terms via a first or second projection is the same term $\pi_1(M)$ or $\pi_2(M)$.

*DMI-LIENS (CNRS URA 1347) Ecole Normale Supérieure - 45, Rue d'Ulm - 75230 Paris France
E-mail:dicosmo@dmi.ens.fr

§INRIA Rocquencourt - Domaine de Voluceau, BP 105 - 78153 Le Chesnay Cedex, France and
CNRS and LRI - Bât 490, Université de Paris-Sud - 91405 Orsay Cedex, France
E-mail:kesner@margaux.inria.fr

These facts can be incorporated in the calculus in the form of equalities, that one can read in at least two different ways:

- *an operational way*: these equalities just state possible *optimizations* of a program. Since a term $\langle \pi_1(M), \pi_2(M) \rangle$ is more complex then M, but behaves the same way, it is convenient to replace all its occurrences by M, as this transformation will yield an equivalent, but more efficient and smaller program. Similarly, we will replace every occurrence of $\lambda x.Mx$ by M.
- *a theoretical way*: these equalities state a relation between a program and its type. They just tell us that whenever a term M has a functional type, then it must really be a function, built by λ-abstraction, so we ought to replace it by $\lambda x.Mx$ if it is not already a function. Similarly, a term M of product type has to be really a pair, built via the pair constructor, or otherwise it must be replaced by $\langle \pi_1(M), \pi_2(M) \rangle$.

As we will briefly see in the Survey, a lot of research activity has focused on the operational reading of these equalities in the tradition of λ-calculus, while only a little on the theoretical one. In this paper we will show how this last reading of the equalities provides a confluent and strongly normalizing reduction system for the simply typed λ-calculus with pairs, sums, unit type (or terminal object) and a bounded recursion operator. We also show that the same reduction system stays confluent when allowing unbounded recursion, while of course loosing the strong normalization property.

2 Survey

Due to the deep connections between λ-calculus, proof theory and category theory, works on extensional equalities have appeared with different motivations in all these fields.

By far, the best known extensional equality is the η axiom that we informally introduced above, written in the λ-calculus formalism as

$$(\eta) \qquad \lambda x.Mx = M \qquad \text{provided } x \text{ is not free in } M$$

This axiom, also known as *extensionality*, has traditionally been turned into a reduction, carrying the same name, by orienting the equality from left to right, interpreting operationally equality as a *contraction*. Such an interpretation is well behaved as it preserves confluence [CF58].

In the early 70's, the attention was focusing on products and the extensional rule for pairs, called *surjective pairing*, which is the analog for product types of the usual η extensional rule.

$$(SP) \qquad \langle \pi_1(M), \pi_2(M) \rangle = M$$

With the previous experience of the η rule, it is easy to understand how, at that time, most of the people thought that the right way to turn such an equality into a rewrite rule was also from left to right, as a contraction. But in 1980, J.W. Klop discovered [Klo80] that, if added to the usual confluent rewrite rules for pure λ-calculus, this interpretation of *SP* breaks confluence[1].

Anyway, this first negative result was shortly after mitigated in [Pot81] for the simply typed λ-calculus with η and SP contractions, by providing a first proof of confluence and strong normalization, later on simplified in different ways (see [Tro86] or [GLT90], for example). From then on, the contraction rule for SP was not considered harmful in a typed framework, until the seminal work by Lambek and Scott [LS86]. There, the decision problem of the equational theory of Cartesian Closed Categories (ccc's) is solved using a particular typed λ-calculus equipped with not only η and

[1] See [Bar84], p. 403-409 for a short history and references.

SP equalities, but also with a special type $\mathbf{T}$ representing the *terminal object* of the ccc's[2]. This distinguished atomic type comes with a further extensional axiom asserting that there is exactly one term $*$ of type $\mathbf{T}$:

$$(Top) \qquad M : \mathbf{T} = *$$

Now, the type $\mathbf{T}$ has the bad property of destroying confluence, if the extensional equalities η and SP are turned into contraction rules: the following are the critical pairs that arise immediately, as first pointed out by Obtulowicz, (see [LS86]):

$$\begin{array}{llll}
\langle *, \pi_2(x)\rangle & {}_{Top}\Leftarrow \quad \langle \pi_1(x), \pi_2(x)\rangle & \Rightarrow_{SP} & x \\
\langle \pi_1(x), *\rangle & {}_{Top}\Leftarrow \quad \langle \pi_1(x), \pi_2(x)\rangle & \Rightarrow_{SP} & x \\
(\lambda x : \mathbf{T}.M*) : \mathbf{T} \to A & {}_{Top}\Leftarrow \quad (\lambda x : \mathbf{T}.Mx) : \mathbf{T} \to A & \Rightarrow_{\eta} & M \\
(\lambda x : A.*) : A \to \mathbf{T} & {}_{Top}\Leftarrow \quad (\lambda x : A.Mx) : A \to \mathbf{T} & \Rightarrow_{\eta} & M
\end{array}$$

It is indeed possible, but not easy, to extend the contractive reduction system in order to recover confluence. A first step towards such a confluent system was taken by Poigné and Voss, who were not inspired by category theory, but by the implementation of algebraic data types [PV87]. In their paper, they study a calculus that includes $\lambda^1\beta\eta\pi*$, and notice that to solve the previous critical pairs one needs to add an infinite number of reduction rules (that can be anyway finitely described). Then confluence of such an extended system can be proved by showing weak confluence and strong normalization. Unfortunately, the critical pair for $(\lambda x : A.Mx) : \mathbf{T} \to A$ is missing there, and the strong normalization proof is incomplete.

More recently, Curien and the first author got interested in a polymorphic extension of $\lambda^1\beta\eta\pi*$, that arose in the study of the theory of object oriented programming and of isomorphisms of types [CDC91]. They give a complete (infinite) set of reduction rules for the calculus, which is proved confluent using just weak confluence, weak normalization and some additional properties.

Meanwhile, in the field of proof theory, Prawitz was suggesting [Pra71] to turn these extensional equalities into *expansion* rules, rather than contractions. Building on such ideas, but motivated by the study of coherence problems in category theory, Mints gives a first faulty proof that in the typed framework *expansion rules*, if handled with care, are weakly normalizing and preserve confluence of the typed calculus [Min79][3].

This idea of using expansion rules seems to have passed unnoticed for a long time, even if the so called η-long normal forms were well known and used in the study of higher order unification problems [Hue76]: only in these last years there has been a renewed interest in expansion rules. In recent work [Jay92], still motivated by category theoretic investigation, Jay explores a simply typed λ-calculus with just $\mathbf{T}$ and a natural number type $\mathbf{N}$ as base types, equipped with an induction combinator for terms of type $\mathbf{N}$. He introduced expansion rules for η and SP that are exactly the same as the ones originally used by Mints, and in [JG92] this calculus is proved confluent and strongly normalizing. Category theory is also the motivation of Cubric [Cub92], who repaired the bug in the original proof by Mints showing confluence and weak normalization (but not strong normalization). Other recent related works are [Dou93], who provides another proof of confluence and strong normalization, and [Aka93], where an interesting divide-and-conquer approach is proposed to prove the same properties.

2.1 Our work

The present paper is inspired by all the previous works, but especially by [Jay92] and [PV87]. We use expansion rules to provide a confluent rewriting system for the typed λ-calculus with not

[2]This is the *Unit* type in languages like ML.

[3]The same idea is present in [Min77].

only products and terminal object, but also sums and recursion. This result is derived from the confluence of a restricted system where recursion is bounded (recursive calls of infinite length are not allowed), which is proved to be weakly confluent and strongly normalizing.

We show that strong normalization of the full system can be reduced to that of the system without expansion rules, for which the traditional techniques can be used. For that purpose, we show that any one step reduction in the calculus with expansions can be *simulated* by a non-empty reduction sequence in the calculus without expansions. It turns out that this result is powerful enough to prove directly also the confluence property, as shown in section 6.

Since the reduction with expansion rules is not a congruence, several fundamental properties that hold for the well known typed λ-calculi have to be reformulated in the expansionary framework in a different way as we will shortly see in Section 4. For this reason we believe that the system with expansion rules deserves to be studied much more carefully, so we will undertake the task of proving directly weak confluence: this will lead us to uncover many of the essential features of this reduction.

We introduce now the calculus and its reduction system in section 3, then we investigate the key properties of the new reduction system: weak confluence (section 4) and strong normalization (section 5). In section 6 we derive the confluence property in two different ways and finally in the conclusion we discuss some further applications of our proof techniques. Due to space limitations, we cannot provide the full proofs, and we refer the interested reader to [DCK93] for full details.

3 The Calculus

It is now time to introduce the calculus we will deal with in this paper. There are two versions, one with bounded recursion, and the other with unbounded recursion, that differ just in the term formation rule and in the equality rule for recursive terms. We will now introduce the calculus with bounded recursion and then describe how the unbounded version can be obtained from it.

3.1 Types and Terms

The set of types of our calculus contains a distinguished type constant $\mathbf{T}$[4], a denumerable set of atomic or base types, and is closed w.r.t. formation of function, product and sum, i.e. if A and B are types, then also $A \to B$, $A \times B$ and $A + B$ are types.

For each type A, we fix a denumerable set of variables of that type. We will use $x, y, z, \ldots$ to range over variables, and for a term M we write $M : A$ to mean that M is a term of type A.

The term formation rules of the calculus can then be presented as follows.

$\Gamma \vdash * : \mathbf{T}$	
$x_1 : A_1, \ldots, x_n : A_n \vdash x_i : A_i \ \ (1 \leq i \leq n)$,	where the x_i's are pairwise distinct
If $\Gamma \vdash M : A \to B$ and $\Gamma \vdash N : A$	then $\Gamma \vdash (MN) : B$
If $\Gamma, x : A \vdash M : B$	then $\Gamma \vdash \lambda x : A.M : A \to B$
If $\Gamma \vdash M : A$ and $\Gamma \vdash N : B$	then $\Gamma \vdash \langle M, N \rangle : A \times B$
If $\Gamma \vdash M : B_1 \times B_2$	then $\Gamma \vdash \pi_i(M) : B_i \ \ (i = 1, 2)$
If $\Gamma \vdash M : B_i$	then $\Gamma \vdash in^i_{B_1+B_2}(M) : B_1 + B_2 \ \ (i = 1, 2)$
If $\Gamma \vdash P : A_1 + A_2$ and $\Gamma \vdash M_i : A_i \to D \ \ (i = 1, 2)$	then $\Gamma \vdash Case(P, M_1, M_2) : D$
If $\Gamma, x : A \vdash M : A$	then $\Gamma \vdash (rec\, x : A.M)^i : A \ \ (i \geq 0)$

Notation 3.1 (Free variables, substitutions) *The set of free variables of a term M will be noted $FV(M)$. We write $[N_1, \ldots, N_n/x_1, \ldots, x_n]$ (often abbreviated $[\overline{N}/\overline{x}]$) for the typed substi-*

[4]This stands for the terminal object in ccc's or for the *Unit* type in languages like ML.

tution mapping each variable $x_i : A_i$ to a term $N_i : A_i$. We write $M[\overline{N}/\overline{x}]$ for the term M where each variable x_i free in M is replaced by N_i.

3.2 Equality

Besides the usual identification of terms up to α conversion (i.e. renaming of bound variables), our calculus is equipped with the following equalities between terms.

$$
\begin{array}{lrcl}
(\beta) & (\lambda x : A.M)N & = & M[N/x] \\
(\pi_1) & \pi_1(\langle M_1, M_2\rangle) & = & M_1 \\
(\pi_2) & \pi_2(\langle M_1, M_2\rangle) & = & M_2 \\
(\rho) & Case(in^1_C(R), M_1, M_2) & = & M_1 R \\
 & Case(in^2_C(R), M_1, M_2) & = & M_2 R
\end{array}
\qquad
\begin{array}{lrcl}
(Top) & M & = & * \text{ if } M : \mathbf{T} \\
(\eta) & \lambda x : A.Mx & = & M \text{ if } \left\{ \begin{array}{l} x \notin FV(M) \\ M : A \to B \end{array} \right. \\
(\delta) & \langle \pi_1(M), \pi_2(M)\rangle & = & M \text{ if } M : A \times B \\
(rec) & (rec\, y : C.M)^{i+1} & = & M[(rec\, y : C.M)^i/y]
\end{array}
$$

The index i that is attached to each *rec* term is a *bound* on the depth of the recursive calls that can originate from it. With such a bound, it is possible to insure the strong normalization of the associated reduction system.

The unbounded system is obtained from the bounded one by simply erasing all the bound indexes from the formation and equality rules (and the associated reduction rules). As we will show later, the bounded system can simulate any finite reduction of the unbounded system, and this fact will make it easy to extend the confluence result for the bounded system to the unbounded one. For simplicity, we will explicitly note the bound index only when necessary, dropping it whenever the properties we discuss hold in both systems.

3.3 The confluent rewriting system

The non extensional equality rules and the rule for $\mathbf{T}$ can be turned into a confluent rewriting system by orienting them from left to right, as follows

$$
\begin{array}{lrcl}
(\beta) & (\lambda x : A.M)N & \longrightarrow & M[N/x] \\
(\pi_i) & \pi_i(\langle M_1, M_2\rangle) & \longrightarrow & M_i, \text{ for } i = 1, 2 \\
(\rho) & Case(in^i_C(R), M_1, M_2) & \longrightarrow & M_i R, \text{ for } i = 1, 2 \\
(rec) & (rec\, y : C.M)^{i+1} & \longrightarrow & M[(rec\, y : C.M)^i/y], \text{ for } i \geq 0 \\
(Top) & M & \longrightarrow & * \text{ if } M : \mathbf{T} \text{ and } M \neq *
\end{array}
$$

But when we want to turn the extensional equalities for functions and pairs into expansions, as explained very clearly by Jay, we must be careful to avoid the following reduction loops:

$$
\begin{array}{lcccl}
\lambda x.M & \rightsquigarrow & \lambda y.(\lambda x.M)y & \rightsquigarrow & \lambda y.M[y/x] =_\alpha \lambda x.M \\
\langle M, N\rangle & \rightsquigarrow & \langle \pi_1(\langle M, N\rangle), \pi_2(\langle M, N\rangle)\rangle & \rightsquigarrow & \langle M, N\rangle \\
MN & \rightsquigarrow & (\lambda x.Mx)N & \rightsquigarrow & MN \\
\pi_i(P) & \rightsquigarrow & \pi_i(\langle \pi_1(P), \pi_2(P)\rangle) & \rightsquigarrow & \pi_i(P)
\end{array}
$$

To break the first two loops we must disallow expansions of terms that are already λ-abstractions or pairs:

$$
\begin{array}{ll}
(\eta) & M \longrightarrow \lambda x : A.Mx \text{ if } \left\{ \begin{array}{l} x \notin FV(M) \\ M : A \to B \text{ and } M \text{ is not a } \lambda\text{-abstraction} \end{array} \right. \\
(\delta) & M \longrightarrow \langle \pi_1(M), \pi_2(M)\rangle \text{ if } \left\{ \begin{array}{l} M : A \times B \text{ and } M \text{ is not a pair} \end{array} \right.
\end{array}
$$

But this is not enough: to break the last two loops we must also forbid the η expansion of a term in a context where this term is applied to an argument, and δ expansion of a term when such

a term is the argument of a projection. This means that we cannot define the one-step reduction relation $\Longrightarrow$ on terms as the least congruence on terms containing the above reductions $\longrightarrow$, as is done usually. Instead, one defines formally $M \Longrightarrow M'$ starting from $\longrightarrow$ by induction on the structure of the term. The definition is the same as a congruence closure but for the two last cases.

Definition 3.2 (One-step reduction)

- *If* $M \longrightarrow M'$*, then* $M \Longrightarrow M'$
- *If* $M \Longrightarrow M'$*, then* $(rec\, x : A.M)^i \Longrightarrow (rec\, x : A.M')^i$

$$\begin{array}{lll} Case(M,N,O) \Longrightarrow Case(M',N,O) & in_C^1(M) \Longrightarrow in_C^1(M') & \langle M,N\rangle \Longrightarrow \langle M',N\rangle \\ Case(N,M,O) \Longrightarrow Case(N,M,'O) & in_C^2(M) \Longrightarrow in_C^2(M') & \langle N,M\rangle \Longrightarrow \langle N,M'\rangle \\ Case(N,O,M) \Longrightarrow Case(N,O,M') & \lambda\, x : A.M \Longrightarrow \lambda\, x : A.M' & NM \Longrightarrow NM' \end{array}$$

- *If* $M \Longrightarrow M'$ *but* $M \xrightarrow{\neg\eta} M'$*, then* $MN \Longrightarrow M'N$
- *If* $M \Longrightarrow M'$ *but* $M \xrightarrow{\neg\delta} M'$*, then* $\pi_i(M) \Longrightarrow \pi_i(M')$ *for* $i = 1, 2$

where $\xrightarrow{\neg\eta}$ *stands for a* $\longrightarrow$ *step that is not an* η *step, and similarly for* δ*.*

Notation 3.3 *The transitive and the reflexive transitive closure of* $\Longrightarrow$ *are noted* $\Longrightarrow^+$ *and* $\Longrightarrow^*$ *respectively. Similarly we define* $\overset{\infty}{\Longrightarrow}$, $\overset{\infty}{\Longrightarrow}^+$ *and* $\overset{\infty}{\Longrightarrow}^*$ *for the unbounded system.*

We will use some standard notions from the theory of rewriting system, such as redex, normal form, confluence, weak confluence, strong normalization, etc, without explicitly redefining them here.

It is also useful to define a notion of *influential positions* of a term: informally, a position in a term is *influential* if it prevents an expansion rule from being applied at the root of the subterm found at that position. For example, M occurs at an influential position in the term MN, as η expansion is forbidden on M, no matters if it is a λ-abstraction or not. Obviously, a position in a term can be influential for η or for δ, but not for both. This notion can be properly formalized, by induction on the structure of the terms, as shown in the full paper.

3.4 Adequacy of expansions for extensional equalities

First of all, it is necessary to show that the limitations imposed on the reduction system do not make us loose any valid equality. We will show that the reduction system just introduced really generates the equalities we defined for the calculus. This comes from the fact that the limitations imposed on the reductions are introduced exactly to avoid reduction loops.

Theorem 3.4 ($\Longrightarrow$ generates E) *The equality* E *and the reflexive, symmetric and transitive closure* R *of* $\Longrightarrow$ *are the same relation.*

Proof.

The fact that R is included in E is evident, as all the reductions rules are derived from the equality axioms by orienting and restricting them.

What we are left to show is $E \subseteq R$. It is enough to show that whenever $M = N$ comes from a single equality axiom, we can either rewrite M to N or N to M (since R is reflexive, symmetric and transitive, the other cases will follow trivially).

The basic idea of the proof is to associate to each of these equality steps a reduction step in R. This is done in the obvious way, except in the cases that would violate one of the restrictions imposed on the expansion rules, which we will solve using exactly the reduction loop that this restriction is supposed to prevent.

Here are the problematic cases and how to deal with them. We use the usual context notation $C[M]$ to indicate a particular occurrence of a subterm M of interest in the term $C[M]$.

- $C[\lambda x.M] =_\eta C[\lambda y.(\lambda x.M)y]$. We cannot associate an η reduction to this equality, as we cannot expand something that is already an abstraction. But we can associate to it a β reduction from $C[\lambda y.(\lambda x.M)y]$ to $C[\lambda y.M[y/x]] = C[\lambda x.M]$.
- $C[\langle M, N\rangle] =_\delta \langle \pi_1(\langle M, N\rangle), \pi_2(\langle M, N\rangle)\rangle]$. We cannot expand something that is already a pair, but we can use the π_i's reduction from $\langle \pi_1(\langle M, N\rangle), \pi_2(\langle M, N\rangle)\rangle]$ to $C[\langle M, N\rangle]$.
- $C[MN] =_\eta C[(\lambda x.Mx)N]$. Here we cannot expand M, which is in an influential position, but again we can use β to go from $C[(\lambda x.Mx)N]$ to $C[MN]$ (recall that $x \notin \mathrm{FV}(M)$).
- $C[\pi_i(P)] =_\delta C[\pi_i(\langle \pi_1(P), \pi_2(P)\rangle)]$. We cannot expand P, but we can use the π_i's to go to $C[\pi_i(P)]$ from $C[\pi_i(\langle \pi_1(P), \pi_2(P)\rangle)]$.

□

This calculus also enjoys the subject reduction property, so reductions preserve types.

Proposition 3.5 (Subject Reduction) *If $\Gamma \vdash R : C$ and $R \Longrightarrow^* R'$, then $\Gamma \vdash R' : C$*

4 Weak Confluence

In this section we set off to prove that the reduction system proposed above is actually weakly confluent, i.e. that whenever $M' \Longleftarrow M \Longrightarrow M''$ we can find a term M''' s.t. $M' \Longrightarrow^* M''' \ {}^*\!\Longleftarrow M''$. The proof is fairly more complex here than in the case of λ-calculus where extensional equalities are interpreted as contractions, and this is due to the fact that the reduction relation $\Longrightarrow$ introduced above *is not a congruence on terms*.

4.1 Some difficulties

In particular, in the simply typed $\lambda-$calculus whenever $M \Longrightarrow^* M'$ then $\pi_i(M) \Longrightarrow^* \pi_i(M')$, and if also $N \Longrightarrow^* N'$, then $MN \Longrightarrow^* M'N'$, but this is no longer true now: indeed, we have $x : A \to B \Longrightarrow \lambda z : A.xz$, but xN cannot reduce to $(\lambda z : A.xz)N$.

These properties still hold for those reduction sequences $M \Longrightarrow^* M'$ that do not involve expansions at the root:

Remark 4.1 *Let $M \equiv M_0 \Longrightarrow M_1 \Longrightarrow \ldots \Longrightarrow M_{n-1} \Longrightarrow M_n \equiv M'$ be a reduction sequence where none of the M_i's is expanded at the root. Then $\pi_i(M) \Longrightarrow^* \pi_i(M')$, for $i = 1, 2$, and, if $N \Longrightarrow^* N'$, then $MN \Longrightarrow^* M'N'$.*

4.2 Solving Critical Pairs

In this calculus, it is no longer true that reduction is stable by substitution, as in the traditional λ-calculus: if $P \Longrightarrow P'$, $N \Longrightarrow N'$, it is not true in general that $P[N/x] \Longrightarrow^* P'[N'/x]$.

Indeed, $x : A \to B \Longrightarrow \lambda z : A.xz$, but $x[\lambda y : A.w/x] = \lambda y : A.w$ cannot reduce in our system to $\lambda z : A.(\lambda y : A.w)z = \lambda z : A.xz[\lambda y : A.w/x]$, and $(yM)[x/y] = xM$ cannot reduce to $(\lambda z : A.xz)M = (yM)[\lambda z : A.xz/y]$.

We can prove some weaker properties: if $P \Longrightarrow P'$, then $P[N/x]$ and $P'[N/x]$ have a common reduct (Lemma 4.2), and similarly $P[N/x]$ and $P[N'/x]$ when $N \Longrightarrow N'$ (Lemma 4.3). This suffices for our purpose of proving weak confluence of the reduction system.

Lemma 4.2 (Substitution Lemma (i))
If $P \Longrightarrow P'$, then $P[N/x] \Longrightarrow^* P'[N/x]$ or $P'[N/x] \Longrightarrow^* P[N/x]$. Moreover, if no expansion take place at the root position of P, then there are no expansions at root positions in the reduction sequences $P[N/x] \Longrightarrow^* P'[N/x]$ and $P'[N/x] \Longrightarrow^* P[N/x]$.

Lemma 4.3 (Substitution Lemma (ii))
If $N \overset{R}{\Longrightarrow} N'$, then $M[N/x] \Longrightarrow^* M'' \ {}^*\!\!\Longleftarrow M[N'/x]$ for some term M''. These reduction sequences contain expansions at the root only if $M \equiv x$ and R is an expansion applied at the root of N.

Example 4.4 *Take $M = \langle xy, x\rangle$, $N = w$ and $N' = \lambda z : A.wz$. Then*

$$M[N/x] = \langle wy, w\rangle \Longrightarrow \langle wy, \lambda z : A.wz\rangle \Longleftarrow \langle(\lambda z : A.wz)y, \lambda z : A.wz\rangle = M[N'/x]$$

Lemma 4.2 and 4.3 suffice to prove that all critical pairs arising from a term M by a β-reduction and another reduction rule can be solved. The other critical pairs are treated in full details in [DCK93]. We can then state the following:

Proposition 4.5 (Critical Pairs are solvable)
If $M \to M'$ and $M \Longrightarrow M''$, then $\exists R$ such that $M' \Longrightarrow^* R$ and $M'' \Longrightarrow^* R$.

4.3 From Solved Critical Pairs to Full Weak Confluence

It is to be noted that the solvability of critical pairs we just proved as Proposition 4.5 does not allow us to deduce the weak confluence of the calculus via the famous Knuth-Bendix Critical Pairs Lemma. That Lemma holds only for algebraic rewrite systems, and not for the λ-calculus, that has the higher order rewrite rule β. We need to prove local confluence explicitly, and to do so the following remark is useful.

Remark 4.6 (Expansion rules) *In case the two reductions $M' \longleftarrow M \Longrightarrow M''$ do not involve η (resp. δ) rules applied at the root positions of M, it is possible to close the diagram without using η (resp. δ) rules at the root, except in the three cases shown below: external π's and internal η, external β and internal δ. Notice that M is not a λ- abstraction in the first diagram, N is not a λ- abstraction in the second and $M[N/x]$ is not a pair in the third one.*

$$\begin{array}{ccc}
\pi_1(\langle M, N\rangle) & \overset{\eta}{\Longrightarrow} & \pi_1(\langle \lambda x.Mx, N\rangle) \\
\pi\downarrow & & \Downarrow\pi \\
M & \xrightarrow{\quad\eta\quad} & \lambda x.Mx
\end{array}
\qquad
\begin{array}{ccc}
\pi_2(\langle M, N\rangle) & \overset{\eta}{\Longrightarrow} & \pi_2(\langle M, \lambda x.Nx\rangle) \\
\pi\downarrow & & \Downarrow\pi \\
N & \xrightarrow{\quad\eta\quad} & \lambda x.Nx
\end{array}$$

$$\begin{array}{ccc}
(\lambda x : A.M)N & \overset{\delta}{\Longrightarrow} & (\lambda x : A.\langle \pi_1(M), \pi_2(M)\rangle)N \\
\beta\downarrow & & \Downarrow\beta \\
M[N/x] & \xrightarrow{\quad\delta\quad} & \langle \pi_1(M[N/x]), \pi_2(M[N/x])\rangle
\end{array}$$

With this additional knowledge, we can prove that $\Longrightarrow$ is actually weakly confluent.

Theorem 4.7 (Weak Confluence) *If $M' \Longleftarrow M \Longrightarrow M''$ then there exist a term M''' such that $M' \Longrightarrow^* M''' \ ^*\Longleftarrow M''$ (i.e. the reduction relation $\Longrightarrow$ is weakly confluent). Furthermore, if the reductions in $M' \Longleftarrow M \Longrightarrow M''$ do not contain η (resp. δ) rules applied at the root of M, it is possible also to close the diagram without applying η (resp. δ) rules at the root, except in the cases shown in the previous Remark 4.6.*

5 Strong Normalization

We provide in this section the proof of strong normalization for our calculus. The key idea is to reduce strong normalization of the system with expansion rules to that of the system without expansion rules and for this, we show how the calculus without expansions can be used to simulate the calculus with expansions. We will use a fundamental property relating strong normalization of two systems:

Proposition 5.1 *Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be two reduction systems and $\mathcal{T}$ a translation from terms in $\mathcal{R}_1$ to terms in $\mathcal{R}_2$. If for every reduction $M_1 \overset{\mathcal{R}_1}{\Longrightarrow} M_2$ there is a non empty reduction sequence $P_1 \overset{\mathcal{R}_2}{\Longrightarrow}{}^+ P_2$ such that $\mathcal{T}(M_i) = P_i$, for $i = 1, 2$, then the strong normalization of $\mathcal{R}_2$ implies that of $\mathcal{R}_1$.*

Proof. Suppose $\mathcal{R}_2$ is strongly normalizing and $\mathcal{R}_1$ is not. Then there is an infinite reduction sequence $M_1 \overset{\mathcal{R}_1}{\Longrightarrow} M_2 \overset{\mathcal{R}_1}{\Longrightarrow} \ldots$ and from this reduction we can construct an infinite reduction sequence $\mathcal{T}(M_1) \overset{\mathcal{R}_2}{\Longrightarrow}{}^+ \mathcal{T}(M_2) \overset{\mathcal{R}_2}{\Longrightarrow}{}^+ \ldots$ which leads to a contradiction. □

The goal is now to find a translation of terms mapping our calculus into itself such that for every possible reduction in the original system from a term M to another term N, there is a reduction sequence from the translation of M to the translation of N, that is *non empty* and *does not* contain any expansion. Then the previous proposition allows us to derive the strong normalization property for the full system from that of the system without expansion rules, which can be proved using standard techniques.

5.1 Simulating Expansions without Expansions

The first naïve idea that comes to the mind is to choose a translation such that expansion rules are completely impossible on a translated term. This essentially amounts to associate to a term M its η-δ normal form, so that translating a term corresponds then to executing all the possible expansions.

Unfortunately, this simple solution is not a good one: if M reduces to N via an expansion, then the translation of M and that of N are the same term, so to such a reduction step in the full system corresponds an *empty* reduction sequence in the translation, and this does not allow us to apply proposition 5.1.

This leads us to consider a more sophisticated translation that maps a term M to a term M° where expansions are not fully executed as above, but just *marked* in such a way that they can be executed during the simulation process, if necessary, by a rule that is not an expansion.

Let us see how to do this on a simple example: take a variable z of type $A_1 \times A_2$, where the A_i's are atomic types different from $\mathbf{T}$. By performing a δ expansion we obtain its normal form w.r.t. expansion rules: $\langle \pi_1(z), \pi_2(z) \rangle$. Instead of executing this reduction, we just *mark* it in the translation by applying to z an appropriate *expansor* term $\lambda x : A_1 \times A_2 . \langle \pi_1(x), \pi_2(x) \rangle$. As for

$\langle \pi_1(z), \pi_2(z) \rangle$, it is in normal form w.r.t. expansions, so the translation does not modify it in any way. Now, we have the reduction sequence

$$z^\circ \equiv (\lambda x : A_1 \times A_2 . \langle \pi_1(x), \pi_2(x) \rangle) z \to_\beta \langle \pi_1(z), \pi_2(z) \rangle$$

where the translation of z reduces to the translation of $\langle \pi_1(z), \pi_2(z) \rangle$, and the δ expansion from z to $\langle \pi_1(z), \pi_2(z) \rangle$ is simulated in the translation by a β-rule. Clearly, in a generic term M there are many positions where an expansion can be performed, so the translation will have to take into account the *structure* of M and insert the appropriate expansors at all these positions.

Anyway, expansors must be carefully defined to correctly represent not only the expansion step arising from a redex already present in M, but also all the expansion sequences that such step can create: if in the previous example the type A_1 is taken to be an arrow type and the type A_2 a product type, then the term $\pi_1(z)$ can be further η-expanded and the term $\pi_2(z)$ can be expanded by a δ-rule, and the expansor $\lambda x : A_1 \times A_2 . \langle \pi_1(x), \pi_2(x) \rangle$ cannot simulate these further possible reductions. This can only be done by storing in the expansor terms all the information on possible future expansions, that is fully contained in the *type* of the term we are marking.

Definition 5.2 (Translation) *To every type C we associate a term, called the expansor of type C and denoted Δ_C, defined by induction as follows:*

$$\begin{array}{ll} \Delta_{A \to B} = & \lambda x : A \to B . \lambda z : A . \Delta_B (x (\Delta_A z)) \\ \Delta_{A \times B} = & \lambda x : A \times B . \langle \Delta_A (\pi_1(x)), \Delta_B (\pi_2(x)) \rangle \\ \Delta_A & \textit{is empty, in any other case} \end{array}$$

We then define a translation M° for a term $M : A$ as follows:

$$M^\circ = \begin{cases} M^{\circ\circ} & \textit{if } M \textit{ is a } \lambda\textit{-abstraction or a pair} \\ \Delta_A^k M^{\circ\circ} \quad \textit{for any } k > 0 & \textit{otherwise} \end{cases}$$

where $\Delta_A^k M$ denotes the term $\underbrace{(\Delta_A \ldots (\Delta_A}_{k \quad times} M) \ldots)$ and $M^{\circ\circ}$ is defined by induction as:

$$\begin{array}{lcllcl} x^{\circ\circ} & = & x & (\lambda x : B . M)^{\circ\circ} & = & \lambda x : B . M^\circ \\ *^{\circ\circ} & = & * & (rec\ y : A . M)^{i\circ\circ} & = & (rec\ y : A . M^\circ)^i \\ \langle M, N \rangle^{\circ\circ} & = & \langle M^\circ, N^\circ \rangle & Case(R, M, N)^{\circ\circ} & = & Case(R^\circ, M^\circ, N^\circ) \\ (MN)^{\circ\circ} & = & (M^{\circ\circ} N^\circ) & \pi_i(M)^{\circ\circ} & = & \pi_i(M^{\circ\circ}) \\ in_C^i(M)^{\circ\circ} & = & in_C^i(M^\circ) & & & \end{array}$$

This corresponds exactly to the marking procedure described before, but for a little detail: in the translation we allow *any* number of markers to be used (the integer k can be any positive number), and not just one as seemed to suffice for the examples above.

The need for this additional twist in the definition is best understood with an example. Consider two atomic types A and B and the term $(\lambda x : A \times B . x) z$: if k is fixed to be one (*i.e.* we allow only one expansor as marker) then its translation $((\lambda x : A \times B . x) z)^\circ$ is $\Delta_{A \times B}((\lambda x : A \times B . \Delta_{A \times B} x) \Delta_{A \times B} z)$. Now $(\lambda x : A \times B . x) z \xrightarrow{\beta} z$, so we have to verify that $((\lambda x : A \times B . x) z)^\circ$ reduces to z° in at least one step. We have:

$$\Delta_{A \times B}((\lambda x : A \times B . \Delta_{A \times B} x) \Delta_{A \times B} z) \Longrightarrow \Delta_{A \times B} \Delta_{A \times B} \Delta_{A \times B} z$$

However, even if both $\Delta^3_{A \times B} z$ and $\Delta_{A \times B} z$ reduce to the same term $\langle \pi_1(z), \pi_2(z) \rangle$, it is not true that $\Delta^3_{A \times B} z \Longrightarrow^* \Delta_{A \times B} z$. Anyway, if we admit $\Delta^3_{A \times B} z$ as a possible translation of z we will have the

desired property relating reductions and translations. Hence, to be precise, our method associates to each term not just one translation, but a whole family of possible translations, all with the same structure, but with different numbers of expansors used as markers.

What is important for our proof is that when we are given a reduction $M_1 \Longrightarrow M_2 \ldots \Longrightarrow M_n$ in the full calculus, then no matter which possible translation M_1° we choose for M_1, the reductions used in the simulation process all go through possible translations M_i° of the M_i.

Translations preserve types and leave unchanged terms where expansions are not possible.

Lemma 5.3 (Type Preservation) *If $\Gamma \vdash M : A$, then $\Gamma \vdash M^\circ : A$ and $\Gamma \vdash M^{\circ\circ} : A$.*

Lemma 5.4 *If M is in normal form or in η-δ normal form, then $M^\circ = M$.*

The next step is to prove that we can apply proposition 5.1 to our system, *i.e,* for every one step reduction from M to N in the full system, there is a non empty reduction sequence in the system without expansions from any translation of M to a translation of N.

The following property is essential to show that every time we perform a β-reduction on a term M in the original system, any translation of M reduces to a translation of the term we have obtained via $\rightarrow_\beta$ from M. Take for example the reduction $(\lambda x : A.M)N \rightarrow_\beta M[N/x]$. We know that $((\lambda x : A.M)N)^\circ = \Delta_A^k((\lambda x : A.M^\circ)N^\circ)$ and we want to show that there is a *non empty* reduction sequence leading to $M[N/x]^\circ$. Since $\Delta_A^k((\lambda x : A.M^\circ)N^\circ) \rightarrow_\beta \Delta_A^k M^\circ[N^\circ/x]$, we have now to check that the term $(M[N/x])^\circ$ can be reached. We state the property as follows:

Lemma 5.5 *If $\Gamma \vdash M : A$, then $\forall k \geq 0$, $\Delta_A^k M^\circ[\overline{N^\circ}/\overline{x}] \Longrightarrow^* (M[\overline{N}/\overline{x}])^\circ$ and no expansions are performed in the reduction sequences.*

Using 5.5 we can show now:

Theorem 5.6 (Simulation) *If $\Gamma \vdash M : A$ and $M \Longrightarrow N$, then $M^\circ \Longrightarrow^+ N^\circ$ and there are no expansions in the reduction sequences.*

5.2 Strong Normalization of the Full Calculus

Having shown that our translation satisfies the hypothesis of Proposition 5.1, all we are now left to prove is that the bounded reduction system without expansion rules is strongly normalizing. This can be established by one of the standard techniques of reducibility, and does not present essential difficulties once the right definitions of *stability* or *reducibility* are given. In the full paper we provide two proofs, one adapting the proof provided by Poigné and Voss in [PV87], and the other adapting Girard's proof from [GLT90]. These standard techniques apply straightforwardly, but the interested reader will nevertheless find in the full paper all the details. It is then finally possible to state the following

Theorem 5.7 (Strong normalization)
The reduction $\Longrightarrow$ for the bounded system with expansions is strongly normalizing.

Proof. By proposition 5.1, theorem 5.6 and the strong normalization of the bounded calculus without expansions . □

6 Confluence of the Full Calculus

We can immediately deduce the confluence property for the bounded system from the weak confluence and strong normalization properties using Newman's Lemma, however, we can also provide an extremely simple and neat proof that does not need the weak confluence property for the expansionary system.

Theorem 6.1 (Confluence) *The relation* $\Longrightarrow$ *is Church-Rosser.*

Proof. Let M be a term s.t. $P_1 \,{}^*\!\Longleftarrow M \Longrightarrow^* P_2$. Since $\Longrightarrow$ is strongly normalizing, we can reduce the terms P_i to their normal forms $\overline{P_i}$. Then we have $\overline{P_1} \,{}^*\!\Longleftarrow M \Longrightarrow^* \overline{P_2}$, and by theorem 5.6 $\overline{P_1}^\circ \,{}^+\!\Longleftarrow M^\circ \Longrightarrow^+ \overline{P_2}^\circ$ without expansions in the reduction sequences. As the system without expansions is confluent (we showed that it is strongly normalizing, and weak confluence without expansions can be shown as easily as for the simply typed lambda calculus), we can close the internal diagram with $\overline{P_1}^\circ \Longrightarrow^* R \,{}^*\!\Longleftarrow \overline{P_2}^\circ$. Now, $\overline{P_i}^\circ =_{lemma\ 5.4} \overline{P_i}$ and therefore we con complete the proof using the reductions $P_1 \Longrightarrow^* \overline{P_1} \Longrightarrow^* R \,{}^*\!\Longleftarrow \overline{P_2} \,{}^*\!\Longleftarrow P_2$ (notice that $\overline{P_1} = R = \overline{P_2}$). □

In order to show confluence of the full calculus we relate in the first place the bounded reduction $\Longrightarrow$ and the unbounded one $\overset{\infty}{\Longrightarrow}$, and then we use the confluence of $\Longrightarrow$ to show the confluence of $\overset{\infty}{\Longrightarrow}$. This very same technique, that originates from early work of Lévy [Lév76], was used in [PV87]. The connection between the reductions $\Longrightarrow$ and $\overset{\infty}{\Longrightarrow}$ comes from the following:

Remark 6.2 *If* $M \Longrightarrow^* N$*, then* $|M| \overset{\infty}{\Longrightarrow} |N|$*, where* $|M|$ *is obtained from* M *by removing all the indices from the* rec *terms.*

Lemma 6.3 *For any reduction sequence* $M_0 \overset{\infty}{\Longrightarrow} M_1 \overset{\infty}{\Longrightarrow} \ldots \overset{\infty}{\Longrightarrow} M_n$*, there exists an indexed computation* $N_0 \Longrightarrow N_1 \Longrightarrow \ldots \Longrightarrow N_n$ *such that* $|N_i| = M_i$*, for* $i = 0 \ldots n$*.*

Confluence of the full calculus results now from the confluence of the bounded calculus.

Theorem 6.4 $\overset{\infty}{\Longrightarrow}$ *is Church Rosser.*

7 Conclusion and Future Work

We have provided a confluent rewriting system for an extensional typed λ-calculus with product, sum, terminal object and recursion, which is also strongly normalizing in case the recursion operator is bounded. There are mainly two relevant technical contributions in this paper: the weak confluence proof and the simulation theorem.

On one hand, let us remark once again that the weak confluence property for a context-sensitive reduction system is not as straightforward as for the reduction systems that are congruencies. The proof is no longer just a matter of a boring but trivial case analysis, so we had to explore and analyze here the fine structure of the reduction system, showing clearly how substitution and reduction interact in the presence of context-sensitive rules.

The simulation theorem, on the other hand, turns out to be the real key tool for this expansionary system: it allows to reduce *both* confluence *and* strong normalization properties to those for the underlying calculus without expansions, that can be proved using the standard techniques. In a sense, this is all that you really need to prove.

It is also important to remark that our techniques can be applied to many other calculi with expansionary rules. For example, we can accommodate in our calculus the weak extensionality for the sum type[5] which is commonly used in proof theory(see [Gir72], for example), namely $Case(P, \lambda x.in^1(x), \lambda y.in^2(y)) = P$. We refer the interested reader to [DCK93] for more details.

[5]The extensional equality for sums is very problematic in its full form, see [Dou90] for a detailed discussion.

References

[Aka93] Yohji Akama. On mints' reductions for ccc-calculus. In *Typed Lambda Calculus and Applications*, number 664 in LNCS. Springer Verlag, 1993.

[Bar84] Henk Barendregt. *The Lambda Calculus; Its syntax and Semantics (revised edition)*. North Holland, 1984.

[CDC91] Pierre-Louis Curien and Roberto Di Cosmo. A confluent reduction system for the λ-calculus with surjective pairing and terminal object. In Leach, Monien, and Artalejo, editors, *Intern. Conf. on Automata, Languages and Programming (ICALP)*, number 510 in LNCS, pages 291–302. Springer-Verlag, 1991.

[CF58] H.B. Curry and R. Feys. *Combinatory Logic*, volume 1. North Holland, 1958.

[Cub92] D. Cubric. On free ccc. Distributed on the `types` mailing list, 1992.

[DCK93] Roberto Di Cosmo and Delia Kesner. Simulating expansions without expansions. Technical report, INRIA, 1993. To appear.

[Dou90] Daniel J. Dougherty. Some reduction properties of a lambda calculus with coproducts and recursive types. Technical report, Wesleyan University, 1990. E-mail: `ddougherty@eagle.wesleyan.edu`.

[Dou93] Daniel J. Dougherty. Some lambda calculi with categorical sums and products. In *Proc. of the Fifth International Conference on Rewriting Techniques and Applications (RTA)*, 1993.

[Gir72] Jean-Yves Girard. *Interprétation fonctionelle et élimination des coupures dans l'arithmétique d'ordre supérieure*. Thèse de doctorat d'état, Université de Paris VII, 1972.

[GLT90] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Cambridge University Press, 1990.

[Hue76] Gérad Huet. Résolution d'équations dans les langages d'ordre $1, 2, \ldots, \omega$. *Thèse d'Etat, Université Paris VII*, 1976.

[Jay92] C. Barry Jay. Long $\beta\eta$ normal forms and confluence (and its revised version). Technical Report ECS-LFCS-91-183, LFCS, University of Edimburgh, 1992.

[JG92] C. Barry Jay and Neil Ghani. The virtues of eta-expansion. Technical Report ECS-LFCS-92-243, LFCS, University of Edimburgh, 1992.

[Klo80] Jan Willem Klop. Combinatory reduction systems. *Mathematical Center Tracts*, 27, 1980.

[Lév76] Jean-Jaques Lévy. An algebraic interpretation of the $\lambda\beta\kappa$-calculus and a labelled λ-calculus. *Theoretical Computer Science*, 2:97–114, 1976.

[LS86] Joachim Lambek and Philip J. Scott. *An introduction to higher order categorical logic*. Cambridge University Press, 1986.

[Min77] Gregory Mints. Closed categories and the theory of proofs. *Zapiski Nauchnykh Seminarov Leningradskogo Otdeleniya Matematicheskogo Instituta im. V.A. Steklova AN SSSR*, 68:83–114, 1977.

[Min79] Gregory Mints. Teorija categorii i teoria dokazatelstv.I. *Aktualnye problemy logiki i metodologii nauky*, pages 252–278, 1979.

[Pot81] Garrel Pottinger. The Church Rosser Theorem for the Typed lambda-calculus with Surjective Pairing. *Notre Dame Journal of Formal Logic*, 22(3):264–268, 1981.

[Pra71] D. Prawitz. Ideas and results in proof theory. *Proceedings of the 2nd Scandinavian Logic Symposium*, pages 235–307, 1971.

[PV87] Axel Poigné and Josef Voss. On the implementation of abstract data types by programming language constructs. *Journal of Computer and System Science*, 34(2-3):340–376, April/June 1987.

[Tro86] Ann S. Troelstra. Strong normalization for typed terms with surjective pairing. *Notre Dame Journal of Formal Logic*, 27(4), 1986.