

A λ -calculus with constructors

Ariel Arbiser¹, Alexandre Miquel², and Alejandro Ríos¹

¹ Departamento de Computación – Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires, Argentina

`{arbiser,rios}@dc.uba.ar`

² PPS & Université Paris 7 – Case 7014, 2 Place Jussieu
75251 PARIS Cedex 05 – France.

`alexandre.miquel@pps.jussieu.fr`

Abstract. We present an extension of the $\lambda(\eta)$ -calculus with a case construct that propagates through functions like a head linear substitution, and show that this construction permits to recover the expressiveness of ML-style pattern matching. We then prove that this system enjoys the Church-Rosser property using a semi-automatic ‘divide and conquer’ technique by which we determine all the pairs of commuting subsystems of the formalism (considering all the possible combinations of the nine primitive reduction rules). Finally, we prove a separation theorem similar to Böhm’s theorem for the whole formalism.

1 Introduction

Lambda-calculus has been introduced by Church in the 30’s [5] as a universal language to express computations of functions. Despite its remarkable simplicity, λ -calculus is rich enough to express all recursive functions. Since the rise of computers, λ -calculus has been used fruitfully as the basis of all functional programming languages, from LISP to the languages of the ML family. From the theoretical point of view, untyped λ -calculus enjoys many good properties [2], such as Church and Rosser’s property expressing determinism of computations. In Logic, λ -calculus is also a fundamental tool to describe the computational contents of proofs via the Curry-Howard correspondence.

Although arbitrarily complex data structures can be encoded in the pure λ -calculus, modern functional programming languages provide primitive constructs for most data structures, for which a purely functional encoding would be inefficient. One of the most popular extensions of λ -calculus is pattern-matching on constructed values (a.k.a. variants), a problem that has been widely investigated in functional programming [11, 8, 12] and in rewriting [14, 6, 4, 10, 9].

However, introducing objects of different kinds—functions and constructed values—in the same formalism addresses the problem of their interaction. What does it mean to apply a constructed value to an argument? Should the constructed value accumulate the extra argument? Or should it produce an error? Similarly, what does it mean to perform case analysis on a function?

Unfortunately, these problems are usually not addressed in the literature because they are irrelevant in a typed setting—applications go with functions,

case analyses with variants. However, one should not forget that one of the reasons of the success of the λ -calculus in computer science and in logic lies in its excellent operational semantics in the untyped case. The best example is given by Böhm’s separation theorem [3] that expresses that two observationally equivalent $\beta\eta$ -normal λ -terms are intentionally equal. In the pure λ -calculus, $\beta\eta$ -normal terms are not canonical forms because they cannot be further reduced; they are canonical forms because the computational behaviour of a $\beta\eta$ -normal term cannot be expressed by another $\beta\eta$ -normal term.

The situation is far from being as clear when we add pattern-matching to the untyped λ -calculus. As far as we know, there is no generalisation of Böhm’s theorem for this kind of extension. One reason for that is that the notion of normal form is not as clear as in the pure λ -calculus, precisely because the traditional operational semantics says nothing about the computational behaviour of ill-typed constructions, such as a case analysis over an abstraction.

An extended operational semantics of case analysis In this paper, we propose an extension of the untyped λ -calculus with constructors and case analysis that fills the holes of the traditional operational semantics. Technically, the main novelty is that we let application and case analysis (written $\{\!\{\theta}\!\}.M$) commute via the (ill-typed¹) reduction rule

$$\text{(CASEAPP)} \quad \{\!\{\theta}\!\}.(MN) \rightarrow (\{\!\{\theta}\!\}.M)N.$$

(Here, θ denotes a *case binding*, that is a finite map from constructors to terms.) Symmetrically, we introduce a reduction rule

$$\text{(CASELAM)} \quad \{\!\{\theta}\!\}.(\lambda x. N) \rightarrow \lambda x. (\{\!\{\theta}\!\}.M) \quad (x \notin FV(\theta))$$

to let case analysis go through abstractions. In this way, case analysis can be understood as a form of head linear explicit substitution. . . of constructors.

Surprisingly, the system we obtain is not only computationally sound—we will show (section 4) that it is confluent and conservative over the untyped $\lambda\eta$ -calculus—but it also permits to decompose ML-style pattern matching (with patterns of any arity) from the construction $\{\!\{\theta}\!\}.M$ that only performs case analysis on constant constructors (section 2).

Finally, we will show (section 5) a theorem of weak separation for the whole calculus, using a separation technique inspired by Böhm’s [3, 2]. For this reason, the formalism provides a special constant written \boxtimes and called the *daimon* (following the terminology and notation of [7]) that requests the termination of the program—something like an `exit` system call—and which will be used as the main technical device to observe normal forms and separate them.

¹ Observe that M is treated as a function in the l.h.s. of the rule whereas it is treated as a constructed value in the r.h.s. This rule should not be confused with the rule of *commutative conversion* $(\{\!\{\theta}\!\}.M)N = \{\!\{\theta}\!\}.MN$ that comes from logic, a rule which is well-typed. . . but incompatible with the reduction rules of our calculus!

2 Syntax and reduction rules

2.1 Syntax

The λ -calculus with constructors distinguishes two kinds of names: *variables* (written x, y, z , etc.) and *constructors* (written c, c' , etc.) The set of variables and the set of constructors are written \mathcal{V} and \mathcal{C} , respectively. In what follows, we assume that both sets \mathcal{V} and \mathcal{C} are denumerable and disjoint.

The terms (written M, N , etc.) and the case bindings (written θ, ϕ , etc.) of the λ -calculus with constructors are inductively defined as follows:

Terms	$M, N ::= x$	(Variable)
	c	(Constructor)
	\boxtimes	(Daimon)
	MN	(Application)
	$\lambda x. M$	(Abstraction)
	$\{\!\{\theta\}\!\}. M$	(Case construct)
Case bindings	$\theta, \phi ::= c_1 \mapsto M_1; \dots; c_n \mapsto M_n$	($c_i \neq c_j$ for $i \neq j$)

We denote the set of terms with $\Lambda_{\mathcal{C}}$, the set of case bindings with \mathcal{B} , and the disjoint union of $\Lambda_{\mathcal{C}}$ and \mathcal{B} with $\Lambda_{\mathcal{C}} + \mathcal{B}$.

Constructor binding Each case binding θ is formed as an finite unordered list of constructor bindings of the form $(c \mapsto M)$ whose l.h.s. are pairwise distinct. We say that a constructor c is *bound* to a term M in a case binding θ if the binding $(c \mapsto M)$ belongs to the list θ . From the definition of case bindings, it is clear that a constructor c is bound to at most one term in a given case binding θ . When there is no such term, we say that the constructor c is *unbound* in θ .

The *size* of a case binding $\theta = (c_1 \mapsto M_1; \dots; c_n \mapsto M_n)$ is written $|\theta|$ and defined by $|\theta| = n$.

We also introduce an (external) operation of *composition* between two case bindings θ and ϕ , which is written $\theta \circ \phi$ and defined by:

$$\theta \circ (c_1 \mapsto M_1; \dots; c_n \mapsto M_n) \equiv c_1 \mapsto \{\!\{\theta\}\!\}. M_1; \dots; c_n \mapsto \{\!\{\theta\}\!\}. M_n$$

(where $\phi \equiv (c_1 \mapsto M_1; \dots; c_n \mapsto M_n)$). Notice that this operation is not syntactically associative, since:

$$(\theta \circ \phi) \circ (c_i \mapsto M_i)_{i=1..n} \equiv (c_i \mapsto \{\!\{\theta \circ \phi\}\!\}. M_i)_{i=1..n}$$

whereas

$$\theta \circ (\phi \circ (c_i \mapsto M_i)_{i=1..n}) \equiv (c_i \mapsto \{\!\{\theta\}\!\}. \{\!\{\phi\}\!\}. M_i)_{i=1..n}$$

However, composition of case bindings only makes sense in the presence of the case conversion reduction rule $\{\!\{\theta\}\!\}. \{\!\{\phi\}\!\}. M \rightarrow \{\!\{\theta \circ \phi\}\!\}. M$ (see 2.2), for which both right-hand sides above are convertible.

Free variables and substitution The notions of bound and free occurrences of a variable are defined as expected. The set of free variables of a term M (resp. a case binding θ) is written $FV(M)$ (resp. $FV(\theta)$). In particular:

$$\begin{aligned} FV((c_i \mapsto M_i)_{i=1..n}) &= FV(M_1) \cup \dots \cup FV(M_n) \\ FV(\{\!\{\theta}\!\}. M) &= FV(\theta) \cup FV(M). \end{aligned}$$

As in the (ordinary) λ -calculus, terms are considered up to α -conversion (i.e. up to a renaming of bound variables). Notice that the renaming policy of the λ -calculus with constructors is strictly the same as in the λ -calculus: it only affects (bound) *variable names*, but leaves *constructor names* unchanged.

The external substitution operation of the λ -calculus, written $M\{x := N\}$, is extended to the λ -calculus with constructors as expected. The same operation is also defined for case bindings (notation: $\theta\{x := N\}$). In particular:

$$\begin{aligned} (c_i \mapsto M_i)_{i=1..n}\{x := N\} &= (c_i \mapsto M_i\{x := N\})_{i=1..n} \\ (\{\!\{\theta}\!\}. M)\{x := N\} &= \{\!\{\theta\{x := N\}\!\}. M\{x := N\}\}. \end{aligned}$$

2.2 Reduction rules

The λ -calculus with constructors has 9 primitive reduction rules that are depicted in Fig. 1.

In what follows, we will be interested not only in the system induced by the 9 reduction rules taken together, but more generally in the subsystems formed by all subsets of these 9 rules. We write $\lambda\mathcal{B}_C$ the calculus generated by all rules of Fig. 1, and \mathcal{B}_C the calculus generated by all rules but `APPLAM` (a.k.a. β).

Notice that `APPLAM` (a.k.a. β) and `LAMAPP` (a.k.a. η) are the only reduction rules that may apply to an ordinary λ -term in $\lambda\mathcal{B}_C$.

2.3 Examples

ML-style pattern-matching Constructors and the case construct of $\lambda\mathcal{B}_C$ basically implement enumerated types. For instance, booleans are represented in $\lambda\mathcal{B}_C$ using two constructors `true` and `false`, and by setting:

$$\text{if } N \text{ then } M_1 \text{ else } M_2 \quad \equiv \quad \{\!\{\text{true} \mapsto M_1; \text{false} \mapsto M_2}\!\}. N$$

From the `CASECONS`-reduction rule of the calculus, we have

$$\begin{aligned} \text{if true then } M_1 \text{ else } M_2 &\equiv \{\!\{\text{true} \mapsto M_1; \text{false} \mapsto M_2}\!\}. \text{true} \rightarrow M_1 \\ \text{if false then } M_1 \text{ else } M_2 &\equiv \{\!\{\text{true} \mapsto M_1; \text{false} \mapsto M_2}\!\}. \text{false} \rightarrow M_2 \end{aligned}$$

However, the `CASEAPP`-reduction rule permits to go beyond simple case analysis, and to implement pattern-matching over non constant patterns. To understand how it works, consider the predecessor function (over unary integers) that maps 0 to 0 and $s\ n$ to n . In $\lambda\mathcal{B}_C$, this function is implemented as

$$\text{pred} \equiv \lambda n. \{\!\{0 \mapsto 0; s \mapsto \lambda z. z}\!\}. n$$

Beta-reduction			
APPLAM	(AL)	$(\lambda x . M)N \rightarrow M\{x := N\}$	
APPDAI	(AD)	$\boxtimes N \rightarrow \boxtimes$	
Eta-reduction			
LAMAPP	(LA)	$\lambda x . Mx \rightarrow M$	$(x \notin FV(M))$
LAMDAl	(LD)	$\lambda x . \boxtimes \rightarrow \boxtimes$	
Case propagation			
CASECONS	(CO)	$\{\!\!\{\theta\}\!\!\}.c \rightarrow M$	$((c \mapsto M) \in \theta)$
CASEDAI	(CD)	$\{\!\!\{\theta\}\!\!\}. \boxtimes \rightarrow \boxtimes$	
CASEAPP	(CA)	$\{\!\!\{\theta\}\!\!\}.(MN) \rightarrow (\{\!\!\{\theta\}\!\!\}.M)N$	
CASELAM	(CL)	$\{\!\!\{\theta\}\!\!\}.\lambda x . M \rightarrow \lambda x . \{\!\!\{\theta\}\!\!\}.M$	$(x \notin FV(\theta))$
Case conversion			
CASECASE	(CC)	$\{\!\!\{\theta\}\!\!\}.\{\!\!\{\phi\}\!\!\}.M \rightarrow \{\!\!\{\theta \circ \phi\}\!\!\}.M$	

Fig. 1. Reduction rules of the λ -calculus with constructors

(where 0 and s are distinct constructors). The computation of $\text{pred } 0$ is obvious from the rules `APPLAM` ($=\beta$) and `CASECONS`:

$$\text{pred } 0 \rightarrow \{\!\!\{0 \mapsto 0; s \mapsto \lambda z . z\}\!\!\}.0 \rightarrow 0.$$

More interesting is the case of $\text{pred } (s N)$ (where N is an arbitrary term)

$$\begin{aligned} \text{pred } (s N) &\rightarrow \{\!\!\{0 \mapsto 0; s \mapsto \lambda z . z\}\!\!\}.(s N) \\ &\rightarrow (\{\!\!\{0 \mapsto 0; s \mapsto \lambda z . z\}\!\!\}.s) N \rightarrow (\lambda z . z) N \rightarrow N \end{aligned}$$

which shows how the case construct captures the head occurrence of the constructor s via the reduction rule `CASEAPP`. More generally, ML-style pattern-matching (on disjoint patterns) is translated in $\lambda\mathcal{B}_C$ as follows:

$$\begin{array}{l} \text{match } N \text{ with} \\ | c_1(x_1, \dots, x_{n_1}) \mapsto M_1 \\ \quad \vdots \\ | c_k(x_1, \dots, x_{n_k}) \mapsto M_k \end{array} \quad \text{becomes} \quad \begin{array}{l} \{\!\!\{c_1 \mapsto \lambda x_1 \cdots x_{n_1} . M_1 ; \\ \quad \vdots \\ c_k \mapsto \lambda x_1 \cdots x_{n_k} . M_k ; \\ \}\!\!\} \cdot N \end{array}$$

Pattern-matching and recursion Recursion is implemented in $\lambda\mathcal{B}_C$ in the same way as in the pure λ -calculus, by using a fixpoint combinator such as Church's Y or Turing's $\Theta \equiv (\lambda y f . f(yy f)) (\lambda y f . f(yy f))$. Combining recursion with case

analysis, one can define recursive functions on algebraic datatypes, such as for example the function ‘append’ that concatenates two lists

$$\text{append} \equiv \Theta (\lambda f l_1 l_2 . \{\text{nil} \mapsto l_2; \text{cons} \mapsto \lambda x l' . \text{cons } x (f l' l_2)\}. l_1)$$

(where lists are represented as usual with two constructors `nil` and `cons`). In what follows, we will use the following arithmetic operators:

$$\begin{aligned} \text{plus} &\equiv \Theta (\lambda f n m . \{0 \mapsto m; s \mapsto \lambda p . s (f p m)\}. n) \\ \text{minus} &\equiv \Theta (\lambda f n p . \{0 \mapsto n; s \mapsto \lambda q . f (\text{pred } n) q\}. p) \end{aligned}$$

Variadic constructors Since constructors have no fixed arity in $\lambda\mathcal{B}_C$, it is tempting to use (some of) them as variadic constructors, that is, as constructors that may be applied to an arbitrary number of arguments.

For instance, a natural idea consists to represent vectors (i.e. variadic tuples) using a single constructor `vec`, letting:

$$(v_1, \dots, v_n) \equiv \text{vec } x_1 \cdots x_n.$$

With such a naive encoding, it is possible to write the concatenation function

$$\text{vappend} \equiv \lambda v_1 v_2 . \{\text{vec} \mapsto v_1\}. v_2,$$

but not the function accessing an element of a vector given by its index. Indeed, accessing the i th element x_i of a vector $v = \text{vec } x_1 \cdots x_n$ requires to skip the first $i - 1$ arguments, but also to drop the $n - i - 1$ remaining arguments. Alas, the latter cannot be implemented in $\lambda\mathcal{B}_C$ since there is no way to count the number of arguments a constructor is applied to.

The solution to fix this problem is to provide the length explicitly, as the first argument of the constructor `vec`, and thus to encode vectors as

$$(v_1, \dots, v_n) \equiv \text{vec } \bar{n} x_1 \cdots x_n.$$

Using this encoding, it is possible to write functions computing the length of a vector or accessing one of its elements:

$$\begin{aligned} \text{skip} &\equiv \Theta (\lambda f n x . \{0 \mapsto x; s \mapsto \lambda pz . f p x\}. n) \\ \text{length} &\equiv \lambda v . \{\text{vec} \mapsto \lambda n . \text{skip } n n\}. v \\ \text{access} &\equiv \lambda v i . \{\text{vec} \mapsto \lambda n . \text{skip } i (\lambda x . \text{skip } (\text{minus } n (s i)) x)\}. v \end{aligned}$$

Of course, it is still possible to define the concatenation function, though its code is slightly more complex than before:

$$\text{vappend} \equiv \lambda v_1 v_2 . \{\text{vec} \mapsto \lambda n_1 . \{\text{vec} \mapsto \lambda n_2 . \text{vec } (\text{plus } n_1 n_2)\}. v_2\}. v_1$$

3 Preliminary results

3.1 Free variables and substitution

We start this section by extending standard lemmas of the λ -calculus to the λ -calculus with constructors.

Lemma 1. *Let M be a term or a case binding, P a term and x a variable. Then $FV(M\{x := P\}) \subset (FV(M) - \{x\}) \cup FV(P)$.*

PROOF: By induction on M . □

Lemma 2. *Let $P, Q \in \mathcal{A}_C + \mathcal{B}$. If $P \rightarrow_{\lambda\mathcal{B}_C} Q$, then $FV(Q) \subseteq FV(P)$.*

PROOF: By induction on P analyzing each one of the rules. Note that the case of rule CASECASE relies on the fact that $FV(\theta \circ \phi) \subset FV(\theta) \cup FV(\phi)$. □

Lemma 3. *Let M be a term or a case binding, P a term, and x a variable such that $x \notin FV(M)$. Then $M\{x := P\} = M$.*

PROOF: By induction on M . □

We can now state the substitution Lemma for $\lambda\mathcal{B}_C$.

Lemma 4 (Substitution lemma). *For all terms and case bindings M , for all terms P, Q and variables x and y such that $x \neq y$ and $x \notin FV(Q)$ we have*

$$M\{x := P\}\{y := Q\} = M\{y := Q\}\{x := P\{y := Q\}\}.$$

PROOF: By induction on M .

- If $M = x$ we have by Lemma 3 that $P\{y := Q\} = P\{y := Q\}$.
- If $M = y$, we have that $Q = Q$ since by hypothesis $x \notin FV(Q)$.
- If $M = z (\neq x, y)$, we have that $z = z$.
- If $M = c$ a constructor, we have that $c = c$.
- If $M = \mathfrak{X}$, we have that $\mathfrak{X} = \mathfrak{X}$.
- If $M = M_1M_2$, $M\{x := P\}\{y := Q\} = M_1\{x := P\}\{y := Q\}M_2\{x := P\}\{y := Q\} =_{IH} M_1\{y := Q\}\{x := P\{y := Q\}\}M_2\{y := Q\}\{x := P\{y := Q\}\} = M\{y := Q\}\{x := P\{y := Q\}\}$.
- If $M = \lambda z.M_1$, $M\{x := P\}\{y := Q\} = \lambda z.M_1\{x := P\}\{y := Q\} =_{IH} \lambda z.M_1\{y := Q\}\{x := P\{y := Q\}\} = M\{y := Q\}\{x := P\{y := Q\}\}$.
- If $M = \theta = (c_i \mapsto M_i)_{i=1..n}$, $\theta\{x := P\}\{y := Q\} = (c_i \mapsto M_i\{x := P\}\{y := Q\})_{i=1..n} =_{IH} (c_i \mapsto M_i\{y := Q\}\{x := P\{y := Q\}\})_{i=1..n} = \theta\{y := Q\}\{x := P\{y := Q\}\}$.
- If $M = \{\theta\}.M_1$, $M\{x := P\}\{y := Q\} = \{\theta\{x := P\}\{y := Q\}\}.M_1\{x := P\}\{y := Q\} =_{IH} \{\theta\{y := Q\}\{x := P\{y := Q\}\}\}.M_1\{y := Q\}\{x := P\{y := Q\}\} = M\{y := Q\}\{x := P\{y := Q\}\}$.

□

3.2 General definitions and commutation results

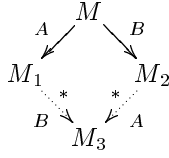
We first recall some classic definitions.

Definition 1. — An Abstract Rewriting System (ARS) is a pair $A = (|A|, \rightarrow_A)$ formed by an arbitrary set $|A|$ (called the carrier of A) equipped with a binary relation \rightarrow_A on $|A|$. We denote by $\rightarrow_A^=$ the reflexive closure of \rightarrow_A , by \rightarrow_A^* the reflexive-transitive closure of \rightarrow_A , and by \simeq_A the reflexive-symmetric-transitive closure of \rightarrow_A .

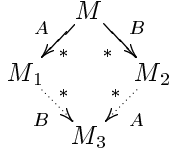
Definition 2. — An ARS A is strongly normalising (SN) if there is no infinite sequence of objects $(M_i)_{i \in \mathbb{N}} \in |A|^{\mathbb{N}}$ such that $M_i \rightarrow_A M_{i+1}$ for all $i \in \mathbb{N}$.

Definition 3. — Let (S, \rightarrow_A) and (S, \rightarrow_B) be two ARSs defined on the same set. We say that:

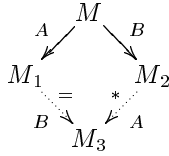
- A weakly commutes with B , written $A //_w B$, if for all M, M_1, M_2 such that $M \rightarrow_A^* M_1$ and $M \rightarrow_B^* M_2$ there exists M_3 such that $M_1 \rightarrow_B^* M_3$ and $M_2 \rightarrow_A^* M_3$. In other words, the following diagram holds:



- A commutes with B , written $A // B$, if for all M, M_1, M_2 such that $M \rightarrow_A^* M_1$ and $M \rightarrow_B^* M_2$ there exists M_3 such that $M_1 \rightarrow_B^* M_3$ and $M_2 \rightarrow_A^* M_3$. In other words, the following diagram holds:



- A strongly commutes with B if for all $M, M_1, M_2 \in S$ such that $M \rightarrow_A M_1$ and $M \rightarrow_B M_2$ there exists $M_3 \in S$ such that $M_1 \rightarrow_{=B} M_3$ and $M_2 \rightarrow_A^* M_3$. In other words, the following diagram holds:



Note that strong commutation is not a symmetrical relation.

Definition 4. Let A be an ARS. We say that

- A is weakly confluent, or weakly Church-Rosser (WCR), if $A //_w A$.
- A is confluent, or Church-Rosser (CR), if $A // A$.
- A enjoys the diamond property if for all M, M_1, M_2 such that $M \rightarrow_A M_1$ and $M \rightarrow_A M_2$ there exists M_3 such that $M_1 \rightarrow_A M_3$ and $M_2 \rightarrow_A M_3$.

Given two ARSs A and B defined on the same carrier set, we write $A + B$ the (set-theoretic) union of both relations. The confluence proof of $\lambda\mathcal{B}_C$ relies on standard results of rewriting [1], and in particular in the following lemmas:

Lemma 5. — *Let A, B, C be ARSs.*

1. *If $A // B$ and $A // C$ then $A // B + C$.*
2. *If $A //_w B$ and $A //_w C$ then $A //_w B + C$.*

PROOF: Both items are immediate from the definition. □

Lemma 6 (Newman's Lemma). — *Let A be an ARS. If A is weakly confluent and strongly normalising, then A is confluent.*

PROOF: See [1] for example. □

In what follows, we will use a generalisation of Newman's lemma to pairs of weakly commuting systems:

Lemma 7. — *If $A //_w B$ and $A + B$ is SN, then $A // B$.*

PROOF: Similar to Newman's Lemma, by a well-founded induction. □

Lemma 8. — *If A strongly commutes with B , then $A // B$.*

PROOF: See [1]. □

3.3 Strong normalization of the \mathcal{B}_C -calculus

We now prove that the sub-calculus $\mathcal{B}_C = (\lambda\mathcal{B}_C \setminus \text{APPLAM})$ enjoys the strong normalization property (SN), a property which will be a key ingredient in the proof of confluence of section 4.

Proposition 1 (SN of \mathcal{B}_C -calculus). *The \mathcal{B}_C -calculus is SN.*

PROOF: Consider the function $h : \mathcal{A}_C + \mathcal{B} \rightarrow \mathbb{N}$ recursively defined by

$$\begin{aligned}
 h(x) &= h(c) = h(\mathbf{\star}) = 1 \\
 h(MN) &= h(M) + h(N) \\
 h(\lambda x.M) &= h(M) + 1 \\
 h(\{\theta\}.M) &= h(\theta) + (|\theta| + 2)h(M) \\
 h((c_i \mapsto M_i)_{i=1..n}) &= \sum_{i=1}^n h(M_i)
 \end{aligned}$$

It is routine to check that h decreases at each \mathcal{B}_C -reduction step. □

Corollary 1 (SN of $\lambda\mathcal{B}_C$ -subsystems). — *For all subsystems s of the $\lambda\mathcal{B}_C$ -calculus (induced by a subset of the 9 primitive rules), s is SN iff $\text{APPLAM} \notin s$.*

4 The Church-Rosser property

4.1 Critical pairs and closure conditions

Each of the 9 primitive reduction rules of $\lambda\mathcal{B}_C$ describes the interaction between two syntactic constructs of the language, which is reflected by the name of the rule: `APPLAM` for ‘Application over a Lambda’, etc. These reduction rules induce 13 different critical pairs, that are summarised in Fig. 2 and 3.

Critical pairs occur for all pairs of rules of the form `FOOBAR`/`BARBAZ`, which corresponds to a situation where a `FOOBAR`-redex and a `BARBAZ`-redex overlap on a syntactic aggregate of the form

$$\text{FOOBAR-redex} \left\{ \begin{array}{c} \text{FOO} \\ | \\ \text{BAR} \\ | \\ \text{BAZ} \end{array} \right\} \text{BARBAZ-redex}$$

A quick examination of Fig. 2 and 3 reveals that each time we have to close such a critical pair, we need to use the third rule `FOOBAZ` when this rule exists. This occurs for the 6 critical pairs (2), (4), (5), (6), (7) and (8) of Fig. 2; in the other cases, the critical pair is closed by the only rules `FOOBAR` and `BARBAZ`.

This remark naturally suggests the following definition:

Definition 5 (Closure conditions). — *We say that a subset s of the 9 rules given in Fig. 1 fulfils the closure conditions and write $s \models \text{CC}$ if:*

(CC1)	$\text{APPLAM} \in s \wedge \text{LAMDAI} \in s \Rightarrow \text{APPDAI} \in s$
(CC2)	$\text{LAMAPP} \in s \wedge \text{APPDAI} \in s \Rightarrow \text{LAMDAI} \in s$
(CC3)	$\text{CASEAPP} \in s \wedge \text{APPLAM} \in s \Rightarrow \text{CASELAM} \in s$
(CC4)	$\text{CASEAPP} \in s \wedge \text{APPDAI} \in s \Rightarrow \text{CASEDAI} \in s$
(CC5)	$\text{CASELAM} \in s \wedge \text{LAMAPP} \in s \Rightarrow \text{CASEAPP} \in s$
(CC6)	$\text{CASELAM} \in s \wedge \text{LAMDAI} \in s \Rightarrow \text{CASEDAI} \in s$

Intuitively, a subset that fulfils the 6 closure conditions defines a system in which all critical pairs can be closed, and thus constitutes a good candidate for Church-Rosser. The aim of this section is to turn this intuition into the

Theorem 1 (Church-Rosser). — *For each of the 512 subsystems s of $\lambda\mathcal{B}_C$ the following propositions are equivalent:*

1. s fulfils the closure conditions (CC1)–(CC6);
2. s is weakly confluent;
3. s is confluent.

Since the full system (i.e. $\lambda\mathcal{B}_C$) obviously fulfils all closure conditions, we will get as an immediate corollary:

Corollary 2 (Church-Rosser). — $\lambda\mathcal{B}_C$ is confluent.

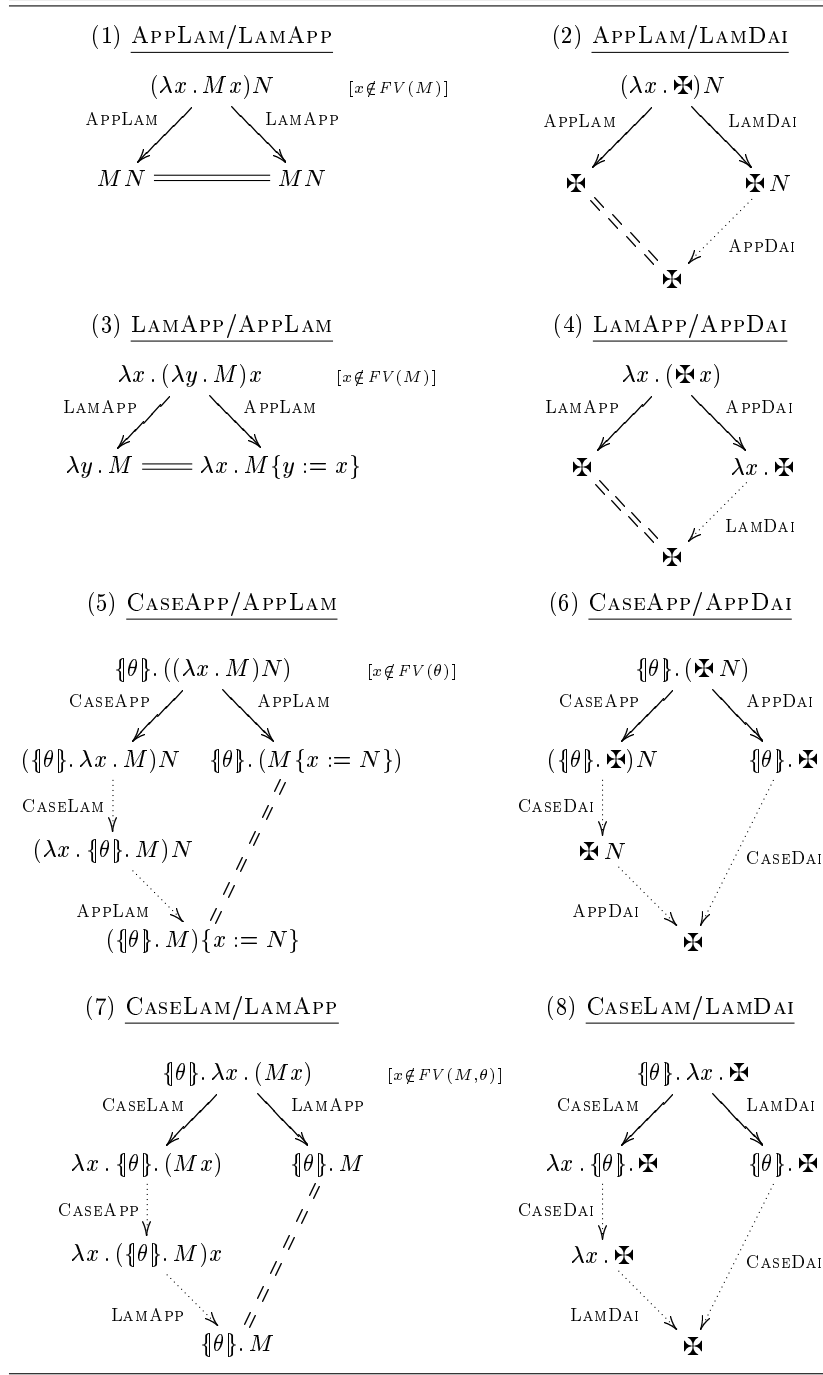


Fig. 2. Critical pairs 1–8 (/13)

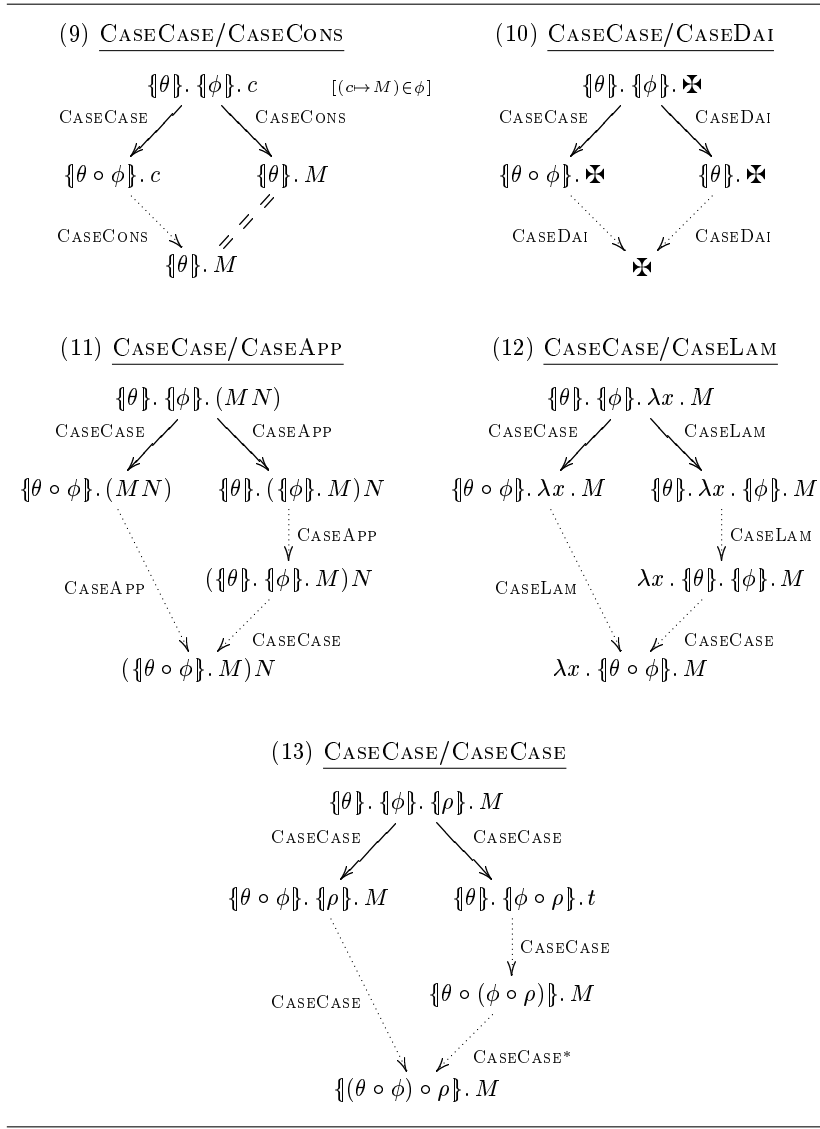


Fig. 3. Critical pairs 9–13 (/13)

The proof of theorem 1 relies on a systematic analysis of the commutation properties of all pairs of subsystems (s_1, s_2) of $\lambda\mathcal{B}_C$. For that, we first have to generalise the notion of closure condition to any pair (s_1, s_2) of subsystems. This leads us to adopt the following definition:

Definition 6 (Binary closure conditions). — *We say that a pair (s_1, s_2) of subsystems fulfils the binary closure conditions and write $(s_1, s_2) \models \text{BCC}$ if*

$$\begin{array}{lll}
(\text{BCC1}) & \text{APPLAM} \in s_1 \wedge \text{LAMDAI} \in s_2 \Rightarrow & \text{APPDAI} \in s_1 \\
(\text{BCC2}) & \text{LAMAPP} \in s_1 \wedge \text{APPDAI} \in s_2 \Rightarrow & \text{LAMDAI} \in s_1 \\
(\text{BCC3}) & \text{CASEAPP} \in s_1 \wedge \text{APPLAM} \in s_2 \Rightarrow & \text{CASELAM} \in s_2 \\
(\text{BCC4}) & \text{CASEAPP} \in s_1 \wedge \text{APPDAI} \in s_2 \Rightarrow & \text{CASEDAI} \in (s_1 \cap s_2) \\
(\text{BCC5}) & \text{CASELAM} \in s_1 \wedge \text{LAMAPP} \in s_2 \Rightarrow & \text{CASEAPP} \in s_2 \\
(\text{BCC6}) & \text{CASELAM} \in s_1 \wedge \text{LAMDAI} \in s_2 \Rightarrow & \text{CASEDAI} \in (s_1 \cap s_2) \\
(\text{BCC7}) & \text{CASECASE} \in s_1 \wedge \text{CASEDAI} \in s_2 \Rightarrow & \text{CASEDAI} \in s_1 \\
(\text{BCC8}) & \text{CASECASE} \in s_1 \wedge \text{CASEAPP} \in s_2 \Rightarrow & \text{CASEAPP} \in s_1 \\
(\text{BCC9}) & \text{CASECASE} \in s_1 \wedge \text{CASELAM} \in s_2 \Rightarrow & \text{CASELAM} \in s_1
\end{array}$$

as well as the 9 symmetric conditions (obtained by exchanging s_1 with s_2).

Again, the 9 binary closure conditions come from an analysis of critical pairs. For example (BCC1) comes from the observation that critical pair (2) of Fig. 2 can be formed as soon as s_1 contains APPLAM and s_2 contains LAMDAI, and that it can be closed only if s_1 contains APPDAI.

We can also remark that when we take $s_1 = s_2 = s$, the binary closure conditions (BCC1)–(BCC6) degenerate to the (simple) closure conditions (CC1)–(CC6) whereas (BCC7)–(BCC9) become tautologies, so that:

Fact 1 — *For all subsystems s of $\lambda\mathcal{B}_C$: $s \models \text{CC}$ iff $(s, s) \models \text{BCC}$.*

We first show that:

Proposition 2. — *For all pairs (s_1, s_2) of subsystems of $\lambda\mathcal{B}_C$ the following propositions are equivalent:*

1. $(s_1, s_2) \models \text{BCC}$ (binary closure conditions);
2. $s_1 //_w s_2$ (weak commutation).

PROOF: (1. \Rightarrow 2.) Let (s_1, s_2) be a pair of subsystems of $\lambda\mathcal{B}_C$ that fulfils the binary closure conditions. By induction of M we show that if $M \rightarrow_{r_1} M_1$ and $M \rightarrow_{r_2} M_2$ (for some $r_1 \in s_1$ and $r_2 \in s_2$), then there is a term M_3 such that $M_1 \rightarrow_{s_2}^* M_3$ and $M_2 \rightarrow_{s_1}^* M_3$. We distinguish cases depending on the structure of M and the rules r_1 and r_2 . The crucial cases correspond to the critical pairs (1)–(13), each of them being closed using the rule(s) given by the corresponding binary closure condition.

($\neg 1. \Rightarrow \neg 2.$) Assume (s_1, s_2) is a pair of subsystems of $\lambda\mathcal{B}_C$ that does not fulfil (at least) one of conditions (BCC1)–(BCC6). We easily build a counter-example from the originating critical pair. \square

4.2 The ‘divide and conquer’ proof technique

Let us now consider the 512×512 matrix formed by all 131,328 (unordered) pairs of subsystems of $\lambda\mathcal{B}_C^2$. From Prop. 2 and Cor. 1, we have:

$$\begin{aligned} s_1 //_w s_2 &\Leftrightarrow (s_1, s_2) \models \text{BCC} \\ (s_1 + s_2) \text{ is SN} &\Leftrightarrow \text{APPLAM} \notin (s_1 + s_2) \end{aligned}$$

for all pairs (s_1, s_2) of subsystems of $\lambda\mathcal{B}_C$. From this it is clear that:

Fact 2 — *Both relations “ $s_1 //_w s_2$ ” and “ $(s_1 + s_2)$ is SN” are decidable.*

With the help of a small computer program, we easily check that:

- There are exactly 13,396 pairs of subsystems (s_1, s_2) such that $s_1 //_w s_2$.
- There are exactly 5,612 pairs of subsystems (s_1, s_2) such that $s_1 //_w s_2$ and $(s_1 + s_2)$ is SN. From lemma 7, we deduce that (at least) all these pairs commute, that is: $s_1 // s_2$.

The situation is summarised in the following table:

	Pairs (s_1, s_2)	$s_1 = s_2$
Commuting + SN (= BCC + \neg APPLAM)	5,612	160 CR+SN
Weakly commuting (= BCC)	13,396	248 WCR
Total	131,328	512

It now remains to show that the commutation property “ $s_1 // s_2$ ” also holds for the remaining $13,396 - 5,612 = 7,784$ pairs of systems that weakly commute, but whose union is not SN.

For that, we propose a simple ‘divide and conquer’ technique to reduce the number of lemmas to prove. The idea is to deduce the remaining 7,784 expected commutation properties from a much smaller set of commutation properties by mechanically applying the second item of lemma 5

$$\text{If } A // B \text{ and } A // C, \text{ then } A // (B + C)$$

to propagate the knowledge of pairs of commuting subsystems (using the algorithm described in Fig. 4). Actually, this method is sufficient to reduce the problem of proving the 7,784 expected commutation properties to the problem of proving the 12 commutation properties of Table 1, since:

Fact 3 — *If the 12 pairs of subsystems of Table 1 commute, then all 13,396 weakly commuting pairs of systems commute.*

PROOF: This is mechanically checked by applying the algorithm of Fig. 4. \square

The rest of this section is thus devoted to the proof of the 12 commutation properties of Table 1.

² In what follows, we count (s_1, s_2) and (s_2, s_1) as a single pair of systems.

(1)	APPLAM // APPLAM
(2)	APPLAM // APPDAI
(3)	APPLAM // LAMAPP
(4)	APPLAM // CASECONS
(5)	APPLAM // CASEDAI
(6)	APPLAM // CASELAM
(7)	APPLAM // CASECASE
(8)	APPLAM + APPDAI // LAMDAI
(9)	APPLAM + APPDAI // LAMAPP + LAMDAI
(10)	APPLAM + CASELAM // CASEAPP
(11)	APPLAM + CASELAM // LAMAPP + CASEAPP
(12)	APPLAM + APPDAI + CASEDAI + CASELAM // LAMAPP + LAMDAI + CASEDAI + CASEAPP

Table 1. The 12 initial commutation lemmas

In the following piece of code, $C[s_1, s_2]$ denotes a symmetric matrix of booleans indexed by all pairs of subsystems (s_1, s_2) . We assume that each assignment $C[s_1, s_2] := b$ implicitly performs the symmetric assignment $C[s_2, s_1] := b$ in order to preserve symmetry.

Global invariant: $C[s_1, s_2] \Rightarrow s_1 // s_2$

```

[Initialise  $C$  with all SN+commuting pairs]
for each  $(s_1, s_2)$  do
   $C[s_1, s_2] := \text{check\_bcc}(s_1, s_2) \wedge \text{APPLAM} \notin (s_1 \cup s_2)$ 
done;

[Manually set the 12 initial commutation lemmas]
for each  $(s_1, s_2) \in \text{Table 1}$  do
   $C[s_1, s_2] := \text{true}$ 
done;

[Close matrix using lemma 5]
while
  there are  $s_1, s_2, s_3$  such that :
   $C[s_1, s_2] \wedge C[s_1, s_3] \wedge \neg C[s_1, (s_2 \cup s_3)]$ 
do  $C[s_1, (s_2 \cup s_3)] := \text{true}$  done;

[Check that all BCC-pairs commute]
for each  $(s_1, s_2)$  do
  assert  $(C[s_1, s_2] \Leftrightarrow \text{check\_bcc}(s_1, s_2))$     [Produces an error if not true]
done
[If no error has been produced, then Fact 3 holds]

```

Fig. 4. Divide an conquer algorithm

Remarks The algorithm of Fig. 4 can be refined in order to determine the smallest possible set of commutation lemmas which is necessary to generate all the desired commutation lemmas from the rule of inference “if $A // B$ and $A // C$, then $A // (B+C)$ ”. (Actually, it can be shown that such a minimal set of commutation lemmas is unique, using a simple combinatorial argument based on the sizes of the sets of rules.) By applying this refined version of the algorithm, we obtain the minimal set of commutation lemmas—which is precisely the set formed by the 12 lemmas of Table 1.

This technique for proving the confluence of a system with many reduction rules is an alternative to the interpretation method [13]—with the benefit that commutation and confluence results are obtained for all subsystems as well.

4.3 Preservation by substitution

We now prove important basic lemmas which relate substitution with the \mathcal{B}_C -rules, to be used in the commutation diagrams of the subsequent lemmas. All these lemmas state that substitution is preserved under the $\lambda\mathcal{B}_C$ -rules.

We formulate these lemmas in a generic way, i.e. treating every calculus rule generically, in order to have a single statement for each one of them.

We will first need the distribution of substitution over composition.

Remark 1. For all case bindings θ, ϕ , for every term P and variable y ,
 $(\theta \circ \phi)\{y := P\} = \theta\{y := P\} \circ \phi\{y := P\}$

PROOF: Let $\phi = (c_i \mapsto M_i)_{i=1, \dots, n}$.
 Then $(\theta \circ \phi)\{y := P\} = (c_i \mapsto \{\theta\}. M_i)_{i=1, \dots, n}\{y := P\}$
 $= (c_i \mapsto \{\theta\{y := P\}\}. M_i\{y := P\})_{i=1, \dots, n}$
 $= \theta\{y := P\} \circ (c_i \mapsto M_i\{y := P\})_{i=1, \dots, n}$
 $= \theta\{y := P\} \circ \phi\{y := P\}$ □

Now we can give the generic

Lemma 9. *Let R be any rule in the set $\{\text{APPLAM}, \text{APPDAI}, \text{LAMAPP}, \text{LAMDAI}, \text{CASELAM}, \text{CASEAPP}, \text{CASEDAI}, \text{CASECONS}, \text{CASECASE}\}$.*

1. *For all terms and case bindings M, N , for every term P and variable y , if $M \rightarrow_R N$ then $M\{y := P\} \rightarrow_R N\{y := P\}$.*
2. *For all terms and case bindings M , for all terms P, Q and variable y , if $P \rightarrow_R Q$, then $M\{y := P\} \xrightarrow{*}_R M\{y := Q\}$*

PROOF:

1. By a straightforward induction on M .
 - If $M = x, y, \lambda, c$, the result holds vacuously.
 - If the reduction is at the root, we have the following possibilities:
 - $R = \text{APPLAM}$, then $M = (\lambda x.Q)R \rightarrow_{\text{APPLAM}} Q\{x := R\} = N$,
 then $M\{y := P\} = (\lambda x.Q\{y := P\})R\{y := P\} \rightarrow_{\text{APPLAM}}$
 $Q\{y := P\}\{x := R\{y := P\}\} =_{L. 4} Q\{x := R\}\{y := P\}$
 $= N\{y := P\}$.

- $R = \text{APPDAI}$, then $M = \mathfrak{X}M_1 \rightarrow_{\text{APPDAI}} \mathfrak{X} = N$,
then $M\{y := P\} = \mathfrak{X}M_1\{y := P\} \rightarrow_{\text{APPDAI}} \mathfrak{X} = N\{y := P\}$.
 - $R = \text{LAMAPP}$, then $M = (\lambda x.Qx) \rightarrow_{\text{LAMAPP}} Q$ where $x \notin FV(Q)$
and $x \neq y$, then $M\{y := P\} = (\lambda x.Q\{y := P\})x$
 $\rightarrow_{\text{LAMAPP}} Q\{y := P\}$ using Lemma 1, since by the free variable
convention $x \notin FV(P)$ and $x \notin FV(Q)$ thus
 $x \notin FV(Q) - \{y\} \cup FV(P)$.
 - $R = \text{LAMDAI}$, then $M = \lambda x.\mathfrak{X} \rightarrow_{\text{LAMDAI}} \mathfrak{X} = N$,
then $M\{y := P\} = \lambda x.\mathfrak{X}\{y := P\} = \lambda x.\mathfrak{X} \rightarrow_{\text{LAMDAI}} \mathfrak{X}$
 $= N\{y := P\}$.
 - $R = \text{CASELAM}$, then $M = \{\theta\}. \lambda x.M_1 \rightarrow_{\text{CASELAM}} \lambda x.\{\theta\}. M_1$
 $= N$, then $M\{y := P\} = \{\theta\}\{y := P\}. (\lambda x.M_1)\{y := P\}$
 $= \{\theta\}\{y := P\}. \lambda x.M_1\{y := P\}$
 $\rightarrow_{\text{CASELAM}} \lambda x.\{\theta\}\{y := P\}. M_1\{y := P\} = N\{y := P\}$.
 - $R = \text{CASEAPP}$, then $M = \{\theta\}. (M_1M_2) \rightarrow_{\text{CASEAPP}} \{\theta\}. M_1M_2 = N$,
then $M\{y := P\} = \{\theta\}\{y := P\}. (M_1\{y := P\}M_2\{y := P\})$
 $\rightarrow_{\text{CASEAPP}} \{\theta\}\{y := P\}. M_1\{y := P\}M_2\{y := P\} = N\{y := P\}$.
 - $R = \text{CASEDAI}$, then $M = \{\theta\}.\mathfrak{X} \rightarrow_{\text{CASEDAI}} \mathfrak{X} = N$,
then $M\{y := P\} = \{\theta\}\{y := P\}.\mathfrak{X}\{y := P\}$
 $= \{\theta\}\{y := P\}.\mathfrak{X} \rightarrow_{\text{CASEDAI}} \mathfrak{X} = N\{y := P\}$.
 - $R = \text{CASECONS}$, then $M = \{(c_i \mapsto M_i)_{i=1,\dots,n}\}. c_j$
 $\rightarrow_{\text{CASECONS}} M_j = N$, then $M\{y := P\}$
 $= \{(c_i \mapsto M_i)_{i=1,\dots,n}\}\{y := P\}. c_j\{y := P\}$
 $= \{(c_i \mapsto M_i\{y := P\})_{i=1,\dots,n}\}. c_j$
 $\rightarrow_{\text{CASECONS}} M_j\{y := P\} = N\{y := P\}$.
 - $R = \text{CASECASE}$, then $M = \{\theta\}.\{\phi\}. Q \rightarrow_{\text{CASECASE}} \{\theta \circ \phi\}. Q = N$,
then $M\{y := P\} = \{\theta\}\{y := P\}.\{\phi\}\{y := P\}. Q\{y := P\}$
 $\rightarrow_{\text{CASECASE}} \{\theta\}\{y := P\} \circ \phi\{y := P\}. Q\{y := P\}$
 $=_{\text{Remark 1}} \{(\theta \circ \phi)\{y := P\}\}. Q\{y := P\} = (\{\theta \circ \phi\}. Q)\{y := P\}$
 $= N\{y := P\}$.
- If $M = M_1M_2 \rightarrow_R N_1M_2$ with $M_1 \rightarrow_R N_1$, then
 $M\{y := P\} = M_1\{y := P\}M_2\{y := P\}$
 $\rightarrow_R^{IH} N_1\{y := P\}M_2\{y := P\} = N\{y := P\}$.
- If $M = M_1M_2 \rightarrow_R N_2M_1$ with $M_2 \rightarrow_R N_2$, analogous to the previous
case.
- If $M = \lambda x.M_1 \rightarrow_R \lambda x.N_1$ with $M_1 \rightarrow_R N_1$, then
 $M\{y := P\} = \lambda x.M_1\{y := P\} \rightarrow_R^{IH} \lambda x.N_1\{y := P\} = N\{y := P\}$ using
the free variable convention.
- If $M = \theta = (c_i \mapsto M_i)_{i=1,\dots,n} \rightarrow_R (c_i \mapsto N_i)_{i=1,\dots,n} = \phi$ where $M_i = N_i$
for $i \neq j$ for some $1 \leq j \leq n$ and $M_j \rightarrow_R N_j$, then
 $M\{y := P\} = (c_i \mapsto M_i\{y := P\})_{i=1,\dots,n}$
 $\rightarrow_R^{IH} (c_i \mapsto N_i\{y := P\})_{i=1,\dots,n} = \theta\{y := P\}$.
- If $M = \{\theta\}. M_1 \rightarrow_R \{\phi\}. M_1 = N$ and $\theta \rightarrow_R \phi$, then
 $M\{y := P\} = \{\theta\}\{y := P\}. M_1\{y := P\}$
 $\rightarrow_R^{IH} \{\phi\}\{y := P\}. M_1\{y := P\}$
 $= N\{y := P\}$.

- If $M = \{\theta\}. M_1 \rightarrow_R \{\theta\}. N_1 = N$ with $M_1 \rightarrow_R N_1$, then

$$M\{y := P\} = \{\theta\}\{y := P\}. M_1\{y := P\}$$

$$\xrightarrow{IH} \{\theta\}\{y := P\}. N_1\{y := P\} = N\{y := P\}.$$
- 2. By induction on M .
 - If $M = y$ we have $P \rightarrow_R Q$ (1 step).
 - If $M = x \neq y$ we have $x \xrightarrow{*}_R x$ (0 steps).
 - If $M = \lambda x$ we have $\lambda x \xrightarrow{*}_R \lambda x$ (0 steps).
 - If $M = c$ we have $c \xrightarrow{*}_R c$ (0 steps).
 - If $M = M_1 M_2$, $M\{y := P\} = M_1\{y := P\} M_2\{y := P\}$

$$\xrightarrow{IH} M_1\{y := Q\} M_2\{y := Q\} = M\{y := Q\}.$$
 - If $M = \lambda x. M_1$, $M\{y := P\} = \lambda x. M_1\{y := P\}$

$$\xrightarrow{IH} \lambda x. M_1\{y := Q\} = M\{y := Q\}.$$
 - If $M = \theta = (c_i \mapsto M_i)_{i=1, \dots, n}$, $M\{y := P\} = (c_i \mapsto M_i\{y := P\})_{i=1, \dots, n}$

$$\xrightarrow{IH} (c_i \mapsto M_i\{y := Q\})_{i=1, \dots, n} = \theta\{y := Q\}.$$
 - If $M = \{\theta\}. M_1$, $M\{y := P\} = \{\theta\}\{y := P\}. M_1\{y := P\}$

$$\xrightarrow{IH} \{\theta\}\{y := Q\}. M_1\{y := Q\} = M\{y := Q\}.$$

□

Actually we need the extension to many-step reductions.

Corollary 3. *Let R be any rule in the set $\{\text{AppLam}, \text{AppDai}, \text{LamApp}, \text{LamDai}, \text{CaseLam}, \text{CaseApp}, \text{CaseDai}, \text{CaseCons}, \text{CaseCase}\}$.*

1. *For all terms and case bindings M, N , for every term P and variable y , if $M \xrightarrow{*}_R N$ then $M\{y := P\} \xrightarrow{*}_R N\{y := P\}$.*
2. *For all terms and case bindings M , for all terms P, Q and variable y , if $P \xrightarrow{*}_R Q$, then $M\{y := P\} \xrightarrow{*}_R M\{y := Q\}$*

PROOF:

1. By Lemma 9(1) and induction on the length of the derivation

$$M \xrightarrow{*}_R N.$$
2. By Lemma 9(2) and induction on the length of the derivation

$$P \xrightarrow{*}_R Q.$$

□

4.4 Commutation of AppLam with AppDai, CaseCons, CaseCase, CaseDai and CaseLam

We prove the necessary commutation lemmas for these rule systems.

Lemma 10. $\text{AppLam} // \text{AppDai}$

PROOF: We prove that the following diagram holds for terms and case bindings (where we denote both sorts by using the letter M):

$$\begin{array}{ccc}
 M & \xrightarrow{AL} & M_1 \\
 AD \downarrow & & AD \downarrow \\
 M_2 & \xrightarrow{=AL} & M_3
 \end{array}$$

It is done by induction on M . There is no critical pair. It uses Corollary 3 (1) and (2). Note that the $=$ subscript appear at the APPLAM -reduction since APPDAI may erase redexes, and a many-step APPDAI -derivation appears because APPLAM may erase or duplicate redexes. Since they strongly commute, by the Commutation Lemma they commute. \square

Lemma 11. $\text{APPLAM} // \text{CASECONS}$

PROOF: We prove that the following diagram holds for terms and case bindings:

$$\begin{array}{ccc}
 M & \xrightarrow{AL} & M_1 \\
 CO \downarrow & & CO \downarrow \\
 M_2 & \xrightarrow{=AL} & M_3
 \end{array}$$

It is done by induction on M . There is no critical pair. It uses Corollary 3 (1) and (2). Note that the $=$ subscript appear at the APPLAM -reduction since CASECONS may erase redexes, and a many-step CASECASE -derivation appears because APPLAM may erase or duplicate redexes. Since they strongly commute, by the Commutation Lemma they commute. \square

Lemma 12. $\text{APPLAM} // \text{CASEDAI}$

PROOF: We prove that the following diagram holds for terms and case bindings:

$$\begin{array}{ccc}
 M & \xrightarrow{AL} & M_1 \\
 CD \downarrow & & CD \downarrow \\
 M_2 & \xrightarrow{=AL} & M_3
 \end{array}$$

It is done by induction on M . There is no critical pair. It uses Corollary 3 (1) and (2). Note that the $=$ subscript appear at the APPLAM -reduction since CASEDAI may erase redexes, and a many-step CASEDAI -derivation appears because APPLAM may erase or duplicate redexes. Since they strongly commute, by the Commutation Lemma they commute. \square

Lemma 13. $\text{APPLAM} // \text{CASELAM}$

PROOF: We prove that the following diagram holds for terms and case bindings:

$$\begin{array}{ccc} M & \xrightarrow{AL} & M_1 \\ CL \downarrow & & CL \downarrow \\ M_2 & \xrightarrow{AL} & M_3 \end{array}$$

It is done by induction on M . There is no critical pair. It uses Corollary 3 (1) and (2). Note that the many-step CASELAM-derivation appears because APPLAM may erase or duplicate redexes. Since they strongly commute, by the Commutation Lemma they commute. \square

4.5 Commutation of AppLam+CaseLam with CaseApp

Lemma 14. *The following diagrams hold for terms and case bindings:*

$$\begin{array}{ccc} M \xrightarrow{CA} M_1 & M \xrightarrow{CA} M_1 & M \xrightarrow{CA} M_1 \\ CL \downarrow & CL \downarrow & CL \downarrow \\ M_2 \xrightarrow{CA} M_3 & M_2 \xrightarrow{CA} M_3 & M_2 \xrightarrow{CA} M_3 \end{array}$$

PROOF:

1. By induction on M . There is no critical pair.
2. By induction on the length of the CA-derivation using item (1).
3. By induction on the length of the CL-derivation using item (1).

\square

Lemma 15. *The following diagram holds for terms and case bindings:*

$$\begin{array}{ccc} M & \xrightarrow{CA} & M_1 \\ AL \downarrow & & =CL \downarrow \\ M_2 & \xrightarrow{CA} & M_3 \\ & & AL \downarrow \end{array}$$

PROOF: By induction on M . There is one critical pair, which closes according to the diagram. It uses Corollary 3 (1) and (2). Note that APPLAM may erase or duplicate a redex thus the many-step CASEAPP-derivation at the bottom. \square

Lemma 16. *The following diagram holds for terms and case bindings:*

$$\begin{array}{ccc} M & \xrightarrow{CA} & M_1 \\ AL \downarrow & & CL \downarrow \\ M_2 & \xrightarrow{CA} & M_3 \\ & & AL \downarrow \end{array}$$

PROOF: By induction on the length of the CASEAPP-derivation, using Lemma 15 and Lemma 14(3). The picture is:

$$\begin{array}{ccccc}
 M & \xrightarrow{CA} & M' & \xrightarrow{CA} & M_1 \\
 \downarrow AL & & \downarrow CL & & \downarrow CL \\
 & & \bullet & \xrightarrow{CA} & \bullet \\
 & & \downarrow AL & & \downarrow =CL \\
 & & & & \bullet \\
 & & & & \downarrow AL \\
 M_2 & \xrightarrow{CA} & \bullet & \xrightarrow{CA} & M_3
 \end{array}$$

where the left rectangle can be closed by IH. \square

Lemma 17. $\text{APPLAM} + \text{CASELAM} // \text{CASEAPP}$

PROOF: We show first that the following diagram holds:

$$\begin{array}{ccc}
 M & \xrightarrow{CA} & M_1 \\
 \downarrow CL+AL & & \downarrow CL+AL \\
 M_2 & \xrightarrow{CA} & M_3
 \end{array}$$

using Lemma 14(3) and Lemma 16. Then we conclude by induction on the derivation $M \xrightarrow{*}_{CL+AL} M_2$. \square

4.6 Commutation of CaseCase with AppLam

To show commutation of CASECASE with APPLAM we will use the parallel reduction technique. We define for this a parallel version of APPLAM, which will also help in proving confluence of the latter.

Definition 7. We define the parallel APPLAM reduction as follows:

$$\begin{array}{c}
 \frac{M \Rightarrow M' \quad N \Rightarrow N'}{(\lambda x.M)N \Rightarrow M'\{x := N'\}} \text{ (pAppLam)} \qquad \frac{}{M \Rightarrow M} \text{ (pRef)} \\
 \\
 \frac{M \Rightarrow M'}{\lambda x.M \Rightarrow \lambda x.M'} \text{ (pLam)} \qquad \frac{M_i \Rightarrow M'_i, i = 1, \dots, n}{(c_i \mapsto M_i)_{i=1, \dots, n} \Rightarrow (c_i \mapsto M'_i)_{i=1, \dots, n}} \text{ (pCB)} \\
 \\
 \frac{M \Rightarrow M' \quad N \Rightarrow N'}{MN \Rightarrow M'N'} \text{ (pApp)} \qquad \frac{M \Rightarrow M' \quad \theta \Rightarrow \theta'}{\{\theta\}.M \Rightarrow \{\theta'\}.M'} \text{ (pCase)}
 \end{array}$$

Now we state Proposition 3 which provides a sequence of easy but useful structural properties to be used afterwards:

Proposition 3 (Structure preservation by AppLam, CaseCase and \Rightarrow).

For all terms M, M_1, M_2, N , case bindings θ, ϕ and variable x

1. if $\lambda x.M \rightarrow_{\text{CASECASE}} N$ then there exists M' such that $N = \lambda x.M'$ with $M \rightarrow_{\text{CASECASE}} M'$
2. if $M_1 M_2 \rightarrow_{\text{CASECASE}} N$ then
 - either there exists M'_1 such that $N = M'_1 M_2$ with $M_1 \rightarrow_{\text{CASECASE}} M'_1$
 - or there exists M'_2 such that $N = M_1 M'_2$ with $M_2 \rightarrow_{\text{CASECASE}} M'_2$
3. if $\theta \rightarrow_{\text{CASECASE}} \phi$ then there exist M_1, \dots, M_n, N , and constructors c_1, \dots, c_n , with $n \geq 1$ such that $\theta = (c_i \mapsto M_i)_{i=1, \dots, n}$ and there exists $1 \leq j \leq n$ such that $\phi = (c_i \mapsto N_i)_{i=1, \dots, n}$ with $N_i = M_i$ for $i \neq j$ and $M_j \rightarrow_{\text{CASECASE}} N_j$
4. if $\{\theta\}.M \rightarrow_{\text{CASECASE}} N$ then
 - either there exists θ' such that $N = \{\theta'\}.M$ with $\theta \rightarrow_{\text{CASECASE}} \theta'$
 - or there exists M' such that $N = \{\theta\}.M'$ with $M \rightarrow_{\text{CASECASE}} M'$
 - or there exists ϕ, M' such that $M = \{\phi\}.M'$ and $N = \{\theta \circ \phi\}.M'$
5. if $\lambda x.M \Rightarrow N$ then there exists M' such that $N = \lambda x.M'$ with $M \Rightarrow M'$
6. if $M_1 M_2 \Rightarrow N$ then
 - either there exist M'_1, M'_2 such that $N = M'_1 M'_2$ with $M_1 \Rightarrow M'_1$ and $M_2 \Rightarrow M'_2$
 - or there exists P, P', M'_2 and a variable y such that $M_1 = \lambda y.P$ and $N = P'\{y := M'_2\}$ with $P \Rightarrow P'$ and $M_2 \Rightarrow M'_2$
7. if $\theta \Rightarrow \phi$ then there exist M_1, \dots, M_n, N , and constructors c_1, \dots, c_n , with $n \geq 1$ such that $\theta = (c_i \mapsto M_i)_{i=1, \dots, n}$ and $\phi = (c_i \mapsto N_i)_{i=1, \dots, n}$ with $M_i \Rightarrow N_i$ for $1 \leq i \leq n$
8. if $\{\theta\}.M \Rightarrow N$ then there exists θ', M' such that $N = \{\theta'\}.M'$ with $\theta \Rightarrow \theta'$ and $M \Rightarrow M'$

PROOF: All items are proved by induction on M or θ , the same way as in the classical λ -calculus. \square

Having defined parallel APPLAM reduction (Definition 7) we will prove that it commutes strongly with CASECASE (Lemma 21), which will imply that APPLAM commutes with CASECASE (Corollary 5). In other words, we will prove that the following diagram holds for terms and case bindings:

$$\begin{array}{ccc}
 M & \xrightarrow{CC} & M_2 \\
 \Downarrow & & \Downarrow \\
 M_1 & \xrightarrow{CC} & M_3
 \end{array}$$

We need the following technical lemmata. All these lemmas will be formulated for Λ_C terms as well as for \mathcal{B} case bindings. Proofs will be by simultaneous induction on terms and case bindings. In what follows, we also use Barendregt's *free variable convention* whenever necessary.

Similar to Lemma 9, substitution is preserved under parallel APPLAM reduction.

Lemma 18. *For every term and case binding M , for all terms P, Q and every variable y , if $P \Rightarrow Q$, then $M\{y := P\} \Rightarrow M\{y := Q\}$*

PROOF: By induction on M .

- If $M = y$, then we have $P \Rightarrow Q$.
- If $M = x \neq y$, then by reflexivity $x \Rightarrow x$.
- If $M = \mathbf{X}$, then by reflexivity $\mathbf{X} \Rightarrow \mathbf{X}$.
- If $M = c$ a constructor, then by reflexivity $c \Rightarrow c$.
- If $M = M_1 M_2$, then by IH and pApp $M\{y := P\} = M_1\{y := P\} M_2\{y := P\} \Rightarrow M_1\{y := Q\} M_2\{y := Q\} = M\{y := Q\}$.
- If $M = \lambda x. M_1$, then by IH and pLam $M\{y := P\} = \lambda x. M_1\{y := P\} \Rightarrow \lambda x. M_1\{y := Q\} = M\{y := Q\}$.
- If $M = \theta = (c_i \mapsto M_i)_{i=1, \dots, n}$, then by IH and (pCB) $M\{y := P\} = (c_i \mapsto M_i\{y := P\})_{i=1, \dots, n} \Rightarrow (c_i \mapsto M_i\{y := Q\})_{i=1, \dots, n} = \theta\{y := Q\}$.
- If $M = \llbracket \theta \rrbracket. N$, then by IH and pCase $M\{y := P\} = \llbracket \theta\{y := P\} \rrbracket. N\{y := P\} \Rightarrow \llbracket \theta\{y := Q\} \rrbracket. N\{y := Q\} = M\{y := Q\}$.

□

The following is a generalization of the previous lemma:

Lemma 19. *For all terms and case bindings P, Q , for all terms R, S and every variable y , if $P \Rightarrow Q$ and $R \Rightarrow S$, then $P\{y := R\} \Rightarrow Q\{y := S\}$*

PROOF: By induction on the derivation of $P \Rightarrow Q$.

- if (pRef) was applied, $P = Q$, then by Lemma 18 $P\{y := R\} \Rightarrow Q\{y := S\}$.
- for (pAppLam), $P = (\lambda x. M)N$, $Q = M'\{x := N'\}$ with $M \Rightarrow M'$ and $N \Rightarrow N'$, then $((\lambda x. M)N)\{y := R\} = (\lambda x. M\{y := R\})N\{y := R\}$. By IH, $M\{y := R\} \Rightarrow M'\{y := S\}$ and $N\{y := R\} \Rightarrow N'\{y := S\}$, thus $(\lambda x. M\{y := R\})N\{y := R\} \Rightarrow M'\{y := S\}\{x := N'\{y := S\}\} =_{L.A} M'\{x := N'\}\{y := S\} = Q\{y := S\}$ since x is fresh by the free variable convention.
- for (pApp), $(MN)\{y := R\} = M\{y := R\}N\{y := R\} \Rightarrow^{IH} M'\{y := S\}N'\{y := S\} = M'N'\{y := S\}$
- for (pLam), $(\lambda x. M)\{y := R\} = \lambda x. M\{y := R\} \Rightarrow^{IH} \lambda x. M'\{y := S\} = (\lambda x. M')\{y := S\}$
- for (pCB), $(c_i \mapsto M_i)_{i=1, \dots, n}\{y := R\} = (c_i \mapsto M_i\{y := R\})_{i=1, \dots, n} \Rightarrow^{IH} (c_i \mapsto M_i'\{y := S\})_{i=1, \dots, n} = (c_i \mapsto M_i')_{i=1, \dots, n}\{y := S\}$
- for (pCase), $\llbracket \theta \rrbracket. N\{y := R\} = \llbracket \theta\{y := R\} \rrbracket. N\{y := R\} \Rightarrow^{IH} \llbracket \theta'\{y := S\} \rrbracket. N'\{y := S\} = \llbracket \theta' \rrbracket. N'\{y := S\}$

□

We still need the following

Lemma 20. *For all case bindings $\theta, \theta', \phi, \phi'$, if $\theta \Rightarrow \theta'$ and $\phi \Rightarrow \phi'$ then $\theta \circ \phi \Rightarrow \theta' \circ \phi'$.*

PROOF: Let $\phi = (d_i \mapsto N_i)_{i=1,\dots,n} \Rightarrow (d_i \mapsto N'_i)_{i=1,\dots,n} = \phi'$ with $N_i \Rightarrow N'_i$ for all $1 \leq i \leq n$. Then $\theta \circ \phi = (d_i \mapsto \{\!\!\{\theta\}\!\!\}. N_i)_{i=1,\dots,n} \Rightarrow (d_i \mapsto \{\!\!\{\theta'\}\!\!\}. N'_i)_{i=1,\dots,n} = \theta' \circ \phi'$. \square

Then we get

Lemma 21 (parallel AppLam and CaseCase strong commutation). *For all terms M, M_1, M_2 , if $M \Rightarrow M_1$ and $M \rightarrow_{\text{CASECASE}} M_2$, then there exists M_3 such that $M_1 \xrightarrow{*}_{\text{CASECASE}} M_3$ and $M_2 \Rightarrow M_3$. And analogously for case bindings. In other words, the following diagrams hold:*

$$\begin{array}{ccc} M & \xrightarrow{CC} & M_2 & \theta & \xrightarrow{CC} & \theta_2 \\ \Downarrow & & \Downarrow & \Downarrow & & \Downarrow \\ M_1 & \xrightarrow{CC} & M_3 & \theta_1 & \xrightarrow{CC} & \theta_3 \end{array}$$

PROOF: We reason by induction on the derivation $M \Rightarrow M_1$. We have the following cases:

1. (pRef) was applied, with $M = M_1$, take $M_3 = M_2$.
2. (pAppLam) was applied, with $M = (\lambda x.P)Q$, $M_1 = P'\{x := Q'\}$, $P \Rightarrow P', Q \Rightarrow Q'$, so by Proposition 3 (2) and (1) we have that either
 - $M_2 = (\lambda x.P'')Q$ with $P \rightarrow_{\text{CASECASE}} P''$, that is

$$\begin{array}{ccc} (\lambda x.P)Q & \xrightarrow{CC} & (\lambda x.P'')Q \\ \Downarrow & & \\ P'\{x := Q'\} & & \end{array}$$

By IH the following diagram holds for some P''' :

$$\begin{array}{ccc} P & \xrightarrow{CC} & P'' \\ \Downarrow & & \Downarrow \\ P' & \xrightarrow{CC} & P''' \end{array}$$

By Corollary 3, $P'\{x := Q'\} \xrightarrow{*}_{\text{CASECASE}} P'''\{x := Q'\}$, and since $P'' \Rightarrow P'''$ and $Q \Rightarrow Q'$, $(\lambda x.P'')Q \Rightarrow P'''\{x := Q'\}$ so the diagram is closed taking $M_3 = P'''\{x := Q'\}$.

- or $M_2 = (\lambda x.P)Q''$, in which case

$$\begin{array}{ccc} (\lambda x.P)Q & \xrightarrow{CC} & (\lambda x.P)Q'' \\ \Downarrow & & \\ P'\{x := Q'\} & & \end{array}$$

By IH the following diagram holds for some Q''' :

$$\begin{array}{ccc} Q & \xrightarrow{CC} & Q'' \\ \Downarrow & & \Downarrow \\ Q' & \xrightarrow{CC} & Q''' \end{array}$$

By Corollary 3, $P'\{x := Q'\} \xrightarrow{*}_{\text{CASECASE}} P'\{x := Q'''\}$, and since $P \Rightarrow P'$ and $Q'' \Rightarrow Q'''$, $(\lambda x.P)Q'' \Rightarrow P'\{x := Q'''\}$ so the diagram is closed taking $M_3 = P'\{x := Q'''\}$.

3. (pLam) was applied, with $M = \lambda x.P$, $M_1 = \lambda x.P''$ with $P \Rightarrow P''$, in which case by Proposition 3 (1) there exists P' such that

$$\begin{array}{ccc} \lambda x.P & \xrightarrow{CC} & \lambda x.P' \\ \Downarrow & & \\ \lambda x.P'' & & \end{array}$$

By IH we have

$$\begin{array}{ccc} P \xrightarrow{CC} P' & \text{then} & \lambda x.P' \\ \Downarrow & & \Downarrow \\ P'' \xrightarrow{CC} P''' & & \lambda x.P'' \xrightarrow{CC} \lambda x.P''' \end{array}$$

4. (pCB) was applied, with $\theta = (c_i \mapsto N_i)_{i=1,\dots,n}$, $\theta'' = (c_i \mapsto N''_i)_{i=1,\dots,n}$, $N_i \Rightarrow N''_i$ for $1 \leq i \leq n$, in which case by Proposition 3 (3)

$$\begin{array}{ccc} \theta & \xrightarrow{CC} & \theta' \\ \Downarrow & & \\ \theta'' & & \end{array}$$

so for some j we have by IH the diagram

$$\begin{array}{ccc} N_j & \xrightarrow{CC} & N'_j \\ \Downarrow & & \Downarrow \\ N''_j & \xrightarrow{CC} & N'''_j \end{array}$$

then taking $\theta''' = (c_i \mapsto N'''_i)_{i=1,\dots,n}$ with $N'''_i = N''_i$ if $i \neq j$

$$\begin{array}{ccc} & & \theta' \\ & & \Downarrow \\ \theta'' & \xrightarrow{CC} & \theta''' \end{array}$$

5. (pApp) was applied, with $M = PQ$, $M_1 = P''Q''$, so by Proposition 3 (2) we have that either
- $M_2 = P'Q$ with $P \rightarrow_{\text{CASECASE}} P'$, in which case

$$\begin{array}{c} PQ \xrightarrow{CC} P'Q \\ \Downarrow \\ P''Q'' \end{array}$$

with $Q \Rightarrow Q''$ and $P \Rightarrow P''$, and by IH we have the diagram

$$\begin{array}{ccc} P \xrightarrow{CC} P' & \text{then} & P'Q \\ \Downarrow & & \Downarrow \\ P'' \xrightarrow{CC} P''' & & P''Q'' \xrightarrow{CC} P'''Q'' \end{array}$$

- or $M_2 = PQ'$ with $Q \rightarrow_{\text{CASECASE}} Q'$, in which case the diagram is closed analogously.
6. (pCase) was applied, with $M = \{\theta\}.Q$, $M_1 = \{\theta'\}.Q'$, $\theta \Rightarrow \theta'$ and $Q \Rightarrow Q'$, so by Proposition 3 (4) we have that either
- CASECASE was applied at the root, i.e. $Q = \{\phi\}.P$, so by Proposition 3 (5) $Q' = \{\phi'\}.P'$ with $\phi \Rightarrow \phi'$, $P \Rightarrow P'$ and we have

$$\begin{array}{c} \{\theta\}.\{\phi\}.P \xrightarrow{CC} \{\theta \circ \phi\}.P \\ \Downarrow \\ \{\theta'\}.\{\phi'\}.P' \end{array}$$

By Lemma 20, $\theta \circ \phi \Rightarrow \theta' \circ \phi'$, thus $\{\theta \circ \phi\}.P \Rightarrow \{\theta' \circ \phi'\}.P'$. Since $\{\theta'\}.\{\phi'\}.P' \rightarrow_{\text{CASECASE}} \{\theta' \circ \phi'\}.P'$, the diagram is closed.

- or an internal CASECASE was applied, then either
 - $\theta \rightarrow_{\text{CASECASE}} \theta''$, so we have

$$\begin{array}{c} \{\theta\}.Q \xrightarrow{CC} \{\theta''\}.Q \\ \Downarrow \\ \{\theta'\}.Q' \end{array}$$

and by IH we have

$$\begin{array}{ccc} \theta \xrightarrow{CC} \theta'' & \text{then} & \{\theta''\}.Q \\ \Downarrow & & \Downarrow \\ \theta' \xrightarrow{CC} \theta''' & & \{\theta'\}.Q' \xrightarrow{CC} \{\theta'''\}.Q' \end{array}$$

- or $Q \rightarrow_{\text{CASECASE}} Q''$, and the diagram is closed analogously.

□

As a generalization of the results in Corollary 3 for APPLAM we have:

Corollary 4. *For all terms and case bindings M, N , for all terms P, Q and variable y , if $M \xrightarrow{*}_{\text{APPLAM}} N$ and $P \xrightarrow{*}_{\text{APPLAM}} Q$, then $M\{y := P\} \xrightarrow{*}_{\text{APPLAM}} N\{y := Q\}$*

PROOF: Since $M \xrightarrow{*}_{\text{APPLAM}} N$, by Corollary 3 (1) we have that $M\{y := P\} \xrightarrow{*}_{\text{APPLAM}} N\{y := P\}$, then by Corollary 3 (2) $M\{y := P\} \xrightarrow{*}_{\text{APPLAM}} N\{y := Q\}$. □

Lemma 22. $\xrightarrow{*}_{\text{APPLAM}} = \xRightarrow{*}$

PROOF: We first show that $\rightarrow_{\text{APPLAM}} \subseteq \Rightarrow$ (it essentially uses the reflexivity rule). For this we show that $M \rightarrow_{\text{APPLAM}} N$ implies $M \Rightarrow N$ by induction on M .

- If $M = x, c, \mathbf{\star}$, the result holds vacuously.
- If the reduction takes place at the root, say $(\lambda x.P)Q \rightarrow_{\text{APPLAM}} P\{x := Q\}$, then $(\lambda x.P)Q \Rightarrow P\{x := Q\}$ using $P \Rightarrow P$ and $Q \Rightarrow Q$ (pRef).
- If $M = PQ \rightarrow_{\text{APPLAM}} P'Q = N$, by IH $P \Rightarrow P'$ so $M \Rightarrow N$ by (pApp).
- If $M = PQ \rightarrow_{\text{APPLAM}} PQ' = N$, analogous.
- If $M = \lambda x.P \rightarrow_{\text{APPLAM}} \lambda x.P' = N$, by IH $P \Rightarrow P'$ so $M \Rightarrow N$ by (pLam).
- If $M = \theta \rightarrow_{\text{APPLAM}} \theta'$, let $\theta = (c_i \mapsto M_i)_{i=1, \dots, n}$ with $M_j \rightarrow_{\text{APPLAM}} M'_j$ for some $1 \leq j \leq n$ and $\theta' = (c_i \mapsto M'_i)_{i=1, \dots, n}$ where $M'_i = M_i$ for $i \neq j$, then by IH $M_j \Rightarrow M'_j$ thus $\theta \Rightarrow \theta'$ by (pCB).
- If $M = \{\theta\}.M_1 \rightarrow_{\text{APPLAM}} \{\theta'\}.M_1 = N$, by IH $\theta \Rightarrow \theta'$ so $M \Rightarrow N$ by (pCase).
- If $M = \{\theta\}.M_1 \rightarrow_{\text{APPLAM}} \{\theta\}.M'_1 = N$, by IH $M_1 \Rightarrow M'_1$ so $M \Rightarrow N$ by (pCase).

Since $\rightarrow_{\text{APPLAM}} \subseteq \Rightarrow$, it follows that $\xrightarrow{*}_{\text{APPLAM}} \subseteq \xRightarrow{*}$.

Now we verify that $\Rightarrow \subseteq \xrightarrow{*}_{\text{APPLAM}}$ (so $\xRightarrow{*} \subseteq \xrightarrow{*}_{\text{APPLAM}}$ will follow). We prove that $P \xrightarrow{*}_{\text{APPLAM}} Q$ by induction on the derivation of $P \Rightarrow Q$:

- if (pRef) was applied, trivial
- for (pAppLam), $P = (\lambda x.M)N \rightarrow_{\text{APPLAM}} M\{x := N\} \xrightarrow{*}_{\text{APPLAM}} M'\{x := N'\} = Q$, where the latter derivation uses Corollary 4, since by IH $M \xrightarrow{*}_{\text{APPLAM}} M'$ and $N \xrightarrow{*}_{\text{APPLAM}} N'$
- for (pApp), $P = MN \xrightarrow{*}_{\text{APPLAM}} M'N' = Q$ since by IH $M \xrightarrow{*}_{\text{APPLAM}} M'$ and $N \xrightarrow{*}_{\text{APPLAM}} N'$
- for (pLam), $P = \lambda x.M \xrightarrow{*}_{\text{APPLAM}} \lambda x.M' = Q$ since by IH $M \xrightarrow{*}_{\text{APPLAM}} M'$
- for (pCB), $P = (c_i \mapsto M_i)_{i=1, \dots, n} \xrightarrow{*}_{\text{APPLAM}} (c_i \mapsto M'_i)_{i=1, \dots, n} = Q$ since by IH $M_i \xrightarrow{*}_{\text{APPLAM}} M'_i$ for $1 \leq i \leq n$
- for (pCase), $P = \{\theta\}.M \xrightarrow{*}_{\text{APPLAM}} \{\theta'\}.M' = Q$ since by IH $\theta \xrightarrow{*}_{\text{APPLAM}} \theta'$ and $M \xrightarrow{*}_{\text{APPLAM}} M'$

□

Now we get

Corollary 5 (AppLam and CaseCase commutation). *For all terms and case bindings M, M_1, M_2 , if $M \xrightarrow{*}_{\text{APPLAM}} M_1$ and $M \xrightarrow{*}_{\text{CASECASE}} M_2$, then there exists M_3 such that $M_1 \xrightarrow{*}_{\text{CASECASE}} M_3$ and $M_2 \xrightarrow{*}_{\text{APPLAM}} M_3$. In other words, the following diagram holds:*

$$\begin{array}{ccc} M & \xrightarrow{CC} & M_2 \\ AL \downarrow & & \downarrow AL \\ M_1 & \xrightarrow{CC} & M_3 \end{array}$$

PROOF: Using Commutation Lemma and Lemma 21 we get that \Rightarrow commutes with $\rightarrow_{\text{CASECASE}}$, therefore by Lemma 22 we conclude. \square

4.7 Commutation of AppLam with AppLam

We can extend the confluence of β -reduction on Λ to the confluence of APPLAM on $\Lambda_C + \mathcal{B}$.

Lemma 23. \Rightarrow *satisfies the diamond property*

PROOF: We prove that the following diagrams hold (respectively for terms and case bindings):

$$\begin{array}{ccc} M \Longrightarrow M_2 & \theta \Longrightarrow \theta_2 \\ \Downarrow & \Downarrow \\ M_1 \dashrightarrow M_3 & \theta_1 \dashrightarrow \theta_3 \end{array}$$

by induction on the derivation of $M \Rightarrow M_1$ (respectively $\theta \Rightarrow \theta_1$). We have the following cases:

1. (pRef) was applied, with $M = M_1$, take $M_3 = M_2$.
2. (pAppLam) was applied, with $M = (\lambda x.P)Q$, $M_1 = P'\{x := Q'\}$ with $P \Rightarrow P'$, $Q \Rightarrow Q'$, so by Proposition 3 (5) and (6) we have that
 - either $M_2 = (\lambda x.P'')Q''$ with $P \Rightarrow P''$, $Q \Rightarrow Q''$, in which case

$$\begin{array}{ccc} (\lambda x.P)Q & \Longrightarrow & (\lambda x.P'')Q'' \\ \Downarrow & & \\ P'\{x := Q'\} & & \end{array}$$

By IH the following diagrams close for some P''' , Q''' :

$$\begin{array}{ccc} P \Longrightarrow P'' & Q \Longrightarrow Q'' \\ \Downarrow & \Downarrow \\ P' \dashrightarrow P''' & Q' \dashrightarrow Q''' \end{array}$$

By Lemma 19, the original diagram is closed taking $M_3 = P'''\{x := Q'''\}$.

– or $M_2 = P''\{x := Q''\}$ with $P \Rightarrow P''$, $Q \Rightarrow Q''$, in which case

$$\begin{array}{c} (\lambda x.P)Q \Longrightarrow P''\{x := Q''\} \\ \Downarrow \\ P'\{x := Q'\} \end{array}$$

By IH the following two diagrams close for some P''' , Q''' :

$$\begin{array}{ccc} P \Longrightarrow P'' & Q \Longrightarrow Q'' & \\ \Downarrow & \Downarrow & \Downarrow \\ P' \cdots \cdots P''' & Q' \cdots \cdots Q''' & \end{array}$$

By Lemma 19, the original diagram is closed taking $M_3 = P'''\{x := Q'''\}$.

3. (pLam) was applied, with $M = \lambda x.P$, $M_1 = \lambda x.P'$, $P \Rightarrow P''$, and by Proposition 3 (5) $M_2 = \lambda x.P'$ with $P \Rightarrow P'$, in which case

$$\begin{array}{c} \lambda x.P \Longrightarrow \lambda x.P' \\ \Downarrow \\ \lambda x.P'' \end{array}$$

By IH we have

$$\begin{array}{ccc} P \Longrightarrow P' & \text{then} & \lambda x.P' \\ \Downarrow & & \Downarrow \\ P'' \cdots \cdots P''' & & \lambda x.P'' \cdots \cdots \lambda x.P''' \end{array}$$

4. (pCB) was applied, with $\theta = (c_i \mapsto N_i)_{i=1,\dots,n}$, $\theta'' = (c_i \mapsto N''_i)_{i=1,\dots,n}$ with $N_i \Rightarrow N''_i$ for $1 \leq i \leq n$ and by Proposition 3 (7) $\theta' = (c_i \mapsto N'_i)_{i=1,\dots,n}$ with $N_i \Rightarrow N'_i$ for $1 \leq i \leq n$, in which case

$$\begin{array}{c} \theta \Longrightarrow \theta' \\ \Downarrow \\ \theta'' \end{array}$$

so for every $1 \leq i \leq n$ we have by IH the diagram

$$\begin{array}{ccc} N_i \Longrightarrow N'_i & & \\ \Downarrow & & \Downarrow \\ N''_i \cdots \cdots N'''_i & & \end{array}$$

then taking $\theta''' = (c_i \mapsto N'''_i)_{i=1,\dots,n}$

$$\begin{array}{c} \theta' \\ \vdots \\ \downarrow \\ \theta'' \dashrightarrow \theta''' \end{array}$$

5. (pApp) was applied, with $M = PQ$, $M_1 = P''Q''$ with $P \Rightarrow P''$, $Q \Rightarrow Q''$, so by Proposition 3 (6) we have that
- either $M_2 = P'Q'$ with $P \Rightarrow P'$, $Q \Rightarrow Q'$, in which case

$$\begin{array}{c} PQ \Longrightarrow P'Q' \\ \Downarrow \\ P''Q'' \end{array}$$

and by IH we have the diagrams

$$\begin{array}{ccc} P \Longrightarrow P' & Q \Longrightarrow Q' & \text{then} \\ \Downarrow & \Downarrow & \\ P'' \dashrightarrow P''' & Q'' \dashrightarrow Q''' & \\ & & P''Q'' \dashrightarrow P'''Q''' \end{array} \quad \begin{array}{c} P'Q' \\ \vdots \\ \downarrow \\ P''Q'' \dashrightarrow P'''Q''' \end{array}$$

- or $M_2 = N'\{x := Q'\}$ with $P = \lambda x.N$, $N \Rightarrow N'$ and $Q \Rightarrow Q'$, in which case by Proposition 3 (5) $P'' = \lambda x.N''$ with $N \Rightarrow N'$. This case is symmetrical with the first item of case 2, so the diagram is closed analogously.
6. (pCase) was applied, with $M = \{\!\!\{\theta\}\!\!\}.Q$, $M_1 = \{\!\!\{\theta''\}\!\!\}.Q''$, so by Proposition 3 (8) we have that $M_2 = \{\!\!\{\theta'\}\!\!\}.P'$ with $\theta \Rightarrow \theta'$, $P \Rightarrow P'$ and

$$\begin{array}{c} \{\!\!\{\theta\}\!\!\}.P \Longrightarrow \{\!\!\{\theta'\}\!\!\}.P' \\ \Downarrow \\ \{\!\!\{\theta''\}\!\!\}.P'' \end{array}$$

and by IH we have

$$\begin{array}{ccc} Q \Longrightarrow Q' & \theta \Longrightarrow \theta' & \text{then} \\ \Downarrow & \Downarrow & \\ Q'' \dashrightarrow Q''' & \theta'' \dashrightarrow \theta''' & \\ & & \{\!\!\{\theta''\}\!\!\}.Q'' \dashrightarrow \{\!\!\{\theta'''\}\!\!\}.Q''' \end{array} \quad \begin{array}{c} \{\!\!\{\theta'\}\!\!\}.Q \\ \vdots \\ \downarrow \\ \{\!\!\{\theta''\}\!\!\}.Q'' \dashrightarrow \{\!\!\{\theta'''\}\!\!\}.Q''' \end{array}$$

□

Corollary 6 (Confluence of AppLam). *APPLAM is confluent.*

PROOF: By previous lemma \Rightarrow satisfies the diamond property and hence it is confluent. We conclude by Lemma 22 that APPLAM is confluent. □

4.8 Commutation of AppLam+AppDai with LamDai

Lemma 24. *The following diagram holds for terms and case bindings:*

$$\begin{array}{ccc}
 M \bullet & \xrightarrow{LD} & \bullet \\
 AL \downarrow & AL+AD & \downarrow \\
 \bullet & \xrightarrow{LD} & \bullet
 \end{array}$$

PROOF: By induction on M . There is only one critical pair, which closes (Figure 3). It uses Corollary 3 (1) and (2). Note that APPLAM may erase or duplicate redexes, thus the $\xrightarrow{*}$ arrow. \square

Lemma 25. *The following diagram holds for terms and case bindings:*

$$\begin{array}{ccc}
 M \bullet & \xrightarrow{LD} & \bullet \\
 AD \downarrow & AD & \downarrow \\
 \bullet & \xrightarrow{=LD} & \bullet
 \end{array}$$

PROOF: By induction on M . There are no critical pairs, thus it closes. Note that APPDAI may erase redexes, thus the = subscript. \square

Lemma 26. $\text{APPLAM} + \text{APPDAI} // \text{LAMDAI}$

PROOF: By induction on M . Use lemmas 24 and 25 to conclude that APPLAM + APPDAI strongly commutes with LAMDAI, therefore they commute. \square

4.9 Commutation of AppLam with LamApp

Just as in the classical λ -calculus, β (here APPLAM) commutes with η (here LAMAPP). To prove strong commutation between both rules we will use Lemma 9 (1) and (2) for LAMAPP.

Lemma 27 (AppLam and LamApp strong commutation). *For all terms and case bindings M, M_1, M_2 , if $M \rightarrow_{\text{LAMAPP}} M_1$ and $M \rightarrow_{\text{APPLAM}} M_2$, then there exists M_3 such that $M_1 \rightarrow_{\text{APPLAM}} M_3$ and $M_2 \xrightarrow{*}_{\text{LAMAPP}} M_3$. In other words, the following diagram holds:*

$$\begin{array}{ccc}
 M & \xrightarrow{LA} & M_1 \\
 AL \downarrow & AL= & \downarrow \\
 M_2 & \xrightarrow{LA} & M_3
 \end{array}$$

PROOF: By induction on M .

- If $M = x, c, \lambda$ it holds vacuously (no redexes).

– for M an application and when the APPLAM-redex occurs at the root, we have the cases:

1. internal LAMAPP at the left

$$\begin{array}{ccc} (\lambda x.P)Q & \xrightarrow{LA} & (\lambda x.P')Q \\ AL \downarrow & & \\ P\{x := Q\} & & \end{array}$$

with $P \rightarrow_{\text{LAMAPP}} P'$, so by Lemma 9 (1),
 $P\{x := Q\} \rightarrow_{\text{LAMAPP}} P'\{x := Q\}$ thus

$$\begin{array}{ccc} & & (\lambda x.P')Q \\ & & \vdots \\ & & AL \downarrow \\ P\{x := Q\} & \xrightarrow{LA} & P'\{x := Q\} \end{array}$$

and the diagram is closed

2. internal LAMAPP at the right

$$\begin{array}{ccc} (\lambda x.P)Q & \xrightarrow{LA} & (\lambda x.P)Q' \\ AL \downarrow & & \\ P\{x := Q\} & & \end{array}$$

with $Q \rightarrow_{\text{LAMAPP}} Q'$, so by Lemma 9 (2),
 $P\{x := Q\} \xrightarrow{*}_{\text{LAMAPP}} P\{x := Q'\}$ thus

$$\begin{array}{ccc} & & (\lambda x.P)Q' \\ & & \vdots \\ & & AL \downarrow \\ P\{x := Q\} & \xrightarrow{LA} & P\{x := Q'\} \end{array}$$

and the diagram is closed

3. the APPLAM-redex overlaps with the LAMAPP-redex, i.e.

$$\begin{array}{ccc} (\lambda x.Px)Q & \xrightarrow{LA} & PQ \\ AL \downarrow & & \\ (Px)\{x := Q\} & & \end{array}$$

with $x \notin FV(P)$, but then $(Px)\{x := Q\} = P\{x := Q\}x\{x := Q\} = PQ$
 so the diagram is closed (in 0 steps).

- For $M = PQ$ and both APPLAM and LAMAPP -reductions internal to P

$$\begin{array}{ccc} PQ & \xrightarrow{LA} & P_1Q \\ AL \downarrow & & \\ P_2Q & & \end{array}$$

by IH there exists P_3 such that

$$\begin{array}{ccc} P & \xrightarrow{LA} & P_1 \\ AL \downarrow & & AL= \downarrow \\ P_2 & \xrightarrow{LA} & P_3 \end{array}$$

thus taking $M_3 = P_3Q$ the diagram is closed

- For $M = PQ$ and both APPLAM and LAMAPP -reductions internal to Q

$$\begin{array}{ccc} PQ & \xrightarrow{LA} & PQ_1 \\ AL \downarrow & & \\ PQ_2 & & \end{array}$$

it is analogous

- For $M = PQ$ and one of the reductions occurs in P and the other in Q , the diagram is one of the following two:

$$\begin{array}{ccc} PQ & \xrightarrow{LA} & PQ_1 \\ AL \downarrow & & \\ P_1Q & & \end{array} \quad \begin{array}{ccc} PQ & \xrightarrow{LA} & P_1Q \\ AL \downarrow & & \\ PQ_1 & & \end{array}$$

and both close to P_1Q_1 in one step

- For $M = \lambda x.P$ and the LAMAPP -redex is internal, we have

$$\begin{array}{ccc} \lambda x.P & \xrightarrow{LA} & \lambda x.P_1 \\ AL \downarrow & & \\ \lambda x.P_2 & & \end{array}$$

so by IH there exists P_3 such that

$$\begin{array}{ccc} P & \xrightarrow{LA} & P_1 \\ AL \downarrow & & AL= \downarrow \\ P_2 & \xrightarrow{LA} & P_3 \end{array}$$

thus taking $M_3 = \lambda x.P_3$ the diagram is closed

- For $M = \{\!\!\{\theta\}\!\!\}.P$ with the APPLAM-redex in P and the LAMAPP-redex in θ , take $M_3 = \{\!\!\{\theta_1\}\!\!\}.P_1$ and the next diagram holds:

$$\begin{array}{ccc} \{\!\!\{\theta\}\!\!\}.P & \xrightarrow{LA} & \{\!\!\{\theta_1\}\!\!\}.P \\ AL \downarrow & & AL \downarrow \\ \{\!\!\{\theta\}\!\!\}.P_1 & \xrightarrow{LA} & \{\!\!\{\theta_1\}\!\!\}.P_1 \end{array}$$

- For $M = \{\!\!\{\theta\}\!\!\}.P$ with the APPLAM-redex in θ and the LAMAPP-redex in P , analogously. □

Now we get

Corollary 7 (AppLam and LamApp commutation). *For all terms and case bindings M, M_1, M_2 , if $M \xrightarrow{*}_{\text{LAMAPP}} M_1$ and $M \xrightarrow{*}_{\text{APPLAM}} M_2$, then there exists M_3 such that $M_1 \xrightarrow{*}_{\text{APPLAM}} M_3$ and $M_2 \xrightarrow{*}_{\text{LAMAPP}} M_3$. In other words, the following diagram holds:*

$$\begin{array}{ccc} M & \xrightarrow{LA} & M_1 \\ AL \downarrow & & AL \downarrow \\ M_2 & \xrightarrow{LA} & M_3 \end{array}$$

PROOF: By the previous lemma they strongly commute, therefore by the Commutation Lemma they commute. □

4.10 Commutation of AppLam+AppDai with LamApp+LamDai

Lemma 28. *The following diagram holds for terms and case bindings:*

$$\begin{array}{ccc} M & \xrightarrow{LA} & M_1 \\ AD \downarrow & & =AD \downarrow \\ M_2 & \xrightarrow{LA+LD} & M_3 \end{array}$$

PROOF: By induction on M . Note that APPDAI may eliminate redexes, thus the $\rightarrow_{=}$ derivation. □

Lemma 29. *The following diagram holds for terms and case bindings:*

$$\begin{array}{ccc} M & \xrightarrow{LA+LD} & M_1 \\ AL+AD \downarrow & & =AL+AD \downarrow \\ M_2 & \xrightarrow{LA+LD} & M_3 \end{array}$$

PROOF: By induction on M , using lemmas 24, 25, 28 and 27. \square

Lemma 30. $\text{APP}_{\text{LAM}} + \text{APP}_{\text{DAI}} // \text{LAM}_{\text{APP}} + \text{LAM}_{\text{DAI}}$

PROOF: Using Lemma 29, they strongly commute, therefore by the Commutation Lemma they commute. \square

4.11 Commutation of $\text{App}_{\text{LAM}} + \text{Case}_{\text{LAM}}$ with $\text{Lam}_{\text{APP}} + \text{Case}_{\text{APP}}$

We begin with auxiliary lemmas.

Lemma 31. *The following diagram holds for terms and case bindings:*

$$\begin{array}{ccc} M & \xrightarrow{\text{LA}} & M_1 \\ \text{CL} \downarrow & & \text{=CL} \downarrow \\ M_2 & \xrightarrow{\text{=CA}} & M_3 \\ & & \text{LA} \downarrow \end{array}$$

PROOF: By induction on M . The interesting cases are the overlaps given by

1. an external CASE_{LAM} -reduction as

$$\begin{array}{ccc} \{\theta\}.(\lambda x.Mx) & \xrightarrow{\text{LA}} & \{\theta\}.M \\ \text{CL} \downarrow & & \parallel \\ \lambda x.(\{\theta\}.(Mx)) & \xrightarrow{\text{=CA}} & \lambda x.(\{\theta\}.M)x \xrightarrow{\text{LA}} \{\theta\}.M \end{array}$$

where $x \notin FV(\theta)$, $x \notin FV(M)$, thus $x \notin FV(\theta) \cup FV(M) = FV(\{\theta\}.M)$ and the diagram holds.

2. an internal CASE_{LAM} -reduction $M \rightarrow_{\text{CASE}_{\text{LAM}}} M'$ as

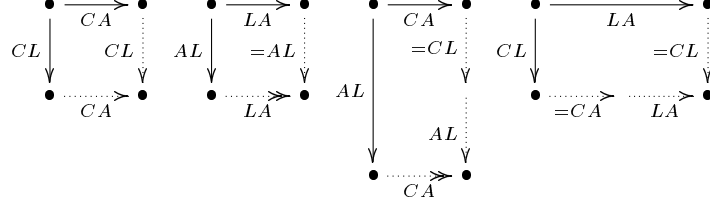
$$\begin{array}{ccc} \lambda x.Mx & \xrightarrow{\text{LA}} & M \\ \text{CL} \downarrow & & \text{CL} \downarrow \\ \lambda x.M'x & \xrightarrow{\text{LA}} & M' \end{array}$$

where $x \notin FV(M)$, and since by Lemma 2 $FV(M') \subseteq FV(M)$, $x \notin FV(M')$ and the diagram holds. \square

Lemma 32. *The following diagram holds for terms and case bindings:*

$$\begin{array}{ccc} M & \xrightarrow{\text{LA+CA}} & M_1 \\ \text{AL+CL} \downarrow & & \text{=CL} \downarrow \\ M_2 & \xrightarrow{\text{LA+CA}} & M_3 \\ & & \text{=AL} \downarrow \end{array}$$

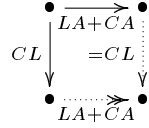
PROOF: We reduce the proof to check the four cases given respectively by the following diagrams:



The first diagram holds by Lemma 14. The second diagram holds by Lemma 27 from last subsection. The third diagram holds by Lemma 15. The fourth diagram holds by Lemma 31. \square

Remark 2. The diagrams in the proof of Lemma 32 show that

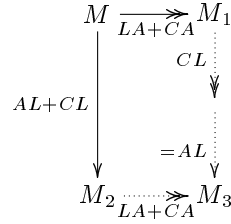
1. LAMAPP+CASEAPP strongly commutes with CASELAM (and hence they commute, by the Commutation Lemma), since the following diagram is a direct consequence of the 1st and 4th diagrams above:



2. CASEAPP strongly commutes with CASELAM.

The first item above will be used in the next lemma:

Lemma 33. *The following diagram holds for terms and case bindings:*



PROOF: The proof is done by induction on the (LAMAPP+CASEAPP)* derivation, using Lemma 32, Remark 2(1) and the Commutation Lemma. We distinguish the following cases:

- If the derivation $M \xrightarrow{*}_{LA+CA} M_1$ has 0 steps, the result is obvious.
- If it has 1 step, it is Lemma 32.
- So let us assume it has ≥ 2 steps. We proceed by lexicographic induction on the pair $(LA + CA + CL\text{-depth of } M, \text{length of the } LA + CA\text{-derivation})$.

The picture is

$$\begin{array}{ccccc}
 M & \xrightarrow{LA+CA} & M'_1 & \xrightarrow{LA+CA} & M_1 \\
 \downarrow AL+CL & & \downarrow CL & & \downarrow CL \\
 & & M'_3 & \xrightarrow{LA+CA} & \bullet \\
 & & \downarrow =AL & & \downarrow CL \\
 & & & & \bullet \\
 & & & & \downarrow =AL \\
 M_2 & \xrightarrow{LA+CA} & M'_2 & \xrightarrow{LA+CA} & M_3
 \end{array}$$

The left rectangle is closed by IH since the second component of the pair decreases. The upper right square can be closed by Remark 2. At M'_3 the lexicographic pair is clearly less than the value at M , since the length of $M \xrightarrow{*}_{LA+CA} M'_1$ is ≥ 1 , so the IH allows to close the lower right rectangle. \square

So finally we get

Lemma 34. $\text{APP LAM} + \text{CASE LAM} // \text{LAM APP} + \text{CASE APP}$

PROOF: Consequence of the previous lemma, using induction on the length of the $\text{APP LAM} + \text{CASE LAM}$ -derivation. \square

4.12 Commutation of $\text{AppLam} + \text{AppDai} + \text{CaseLam} + \text{CaseDai}$ with $\text{LamApp} + \text{LamDai} + \text{CaseApp} + \text{CaseDai}$

For simplicity, let us call

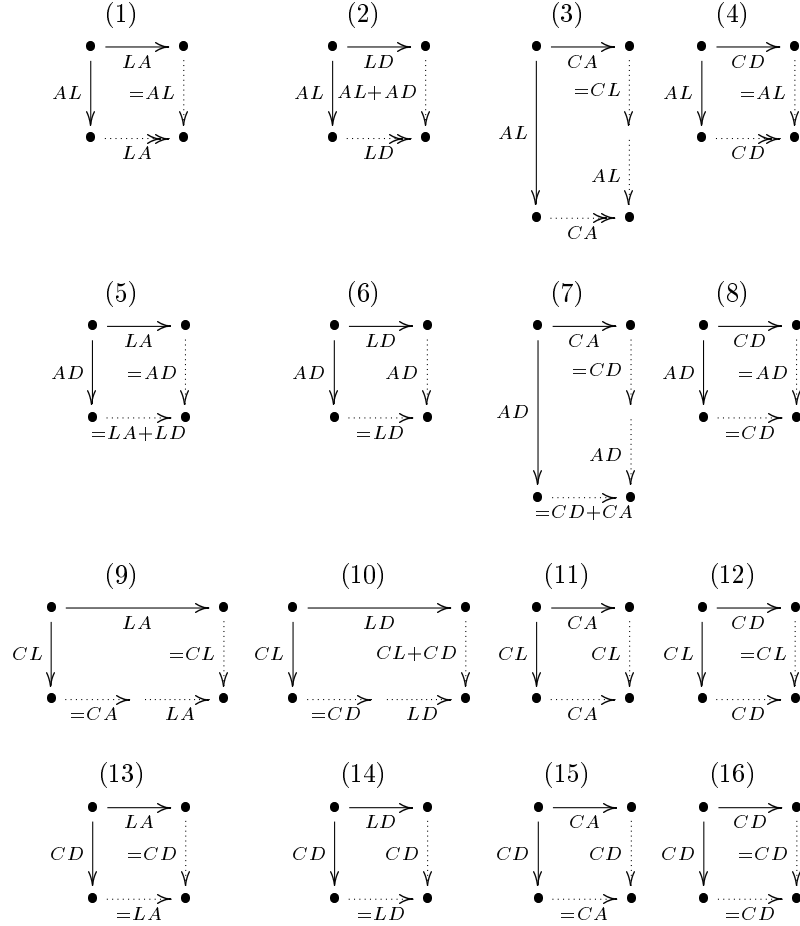
- $S_1 = \text{APP LAM} + \text{APP DAI} + \text{CASE LAM} + \text{CASE DAI}$
- $S_2 = \text{LAM APP} + \text{LAM DAI} + \text{CASE APP} + \text{CASE DAI}$.

Then we can give the

Lemma 35. *The following diagram holds for terms and case bindings:*

$$\begin{array}{ccc}
 M & \xrightarrow{S_2} & M_1 \\
 \downarrow S_1 & \text{=} & \downarrow CL+CD \\
 & & \downarrow \\
 & & \text{=} & \downarrow S_1 \\
 M_2 & \xrightarrow{S_2} & M_3
 \end{array}$$

PROOF: We reduce the proof to check the different cases. In this proof, when there is no critical pair, the diagram can be closed with a single step, noting that certain reduction steps can eventually erase the other redex (thus the presence of “=” subscripts in those cases), and also an APPLAM-reduction step may erase or duplicate the other redex (thus the presence of a $\overset{*}{\rightarrow}$ arrow). For the cases where there is a critical pair, we refer the reader to Figures 2 and 3. We have sixteen cases given respectively by the following diagrams:



Remarks for the preceding diagrams:

1. use Lemma 27 to close, note that AL may erase or duplicate the LA-redex
2. one critical pair, AL may erase or duplicate the LD-redex
3. one critical pair, AL may erase or duplicate the CA-redex
4. no critical pair, AL may erase or duplicate the CD-redex, CD may erase the AL-redex
5. one critical pair, AD may erase the LA-redex and LA may erase or duplicate the AD-redex

6. no critical pair, AD may erase the LD-redex
7. one critical pair, AD may erase the CA-redex
8. no critical pair, AD may erase the CD-redex and CD may erase the AD-redex
9. one critical pair, LA may erase the CD-redex
10. one critical pair, these rules do not erase nor duplicate redexes
11. no critical pair, these rules do not erase nor duplicate redexes
12. no critical pair, CD may erase the CL-redex
13. no critical pair, LA may erase the CD-redex and CD may erase the LA-redex
14. no critical pair, CD may erase the LD-redex
15. no critical pair, CD may erase the CA-redex
16. no critical pair, one CD may erase the other CD-redex

□

Lemma 36. *The following diagram holds for terms and case bindings:*

$$\begin{array}{ccc}
 M & \xrightarrow{S_2} & M_1 \\
 S_1 \downarrow & & \downarrow S_1 \\
 M_2 & \xrightarrow{S_2} & M_3
 \end{array}$$

PROOF: By well-founded induction on the $S_2 + \text{CASELAM} + \text{CASEDAI}$ -depth of M . If the length of the S_2 -derivation is 0, trivial. If it is 1, use Lemma 35. Else, the picture is:

$$\begin{array}{ccccc}
 M & \xrightarrow{S_2} & M' & \xrightarrow{S_2} & \bullet \\
 \downarrow S_1 & \text{=CL+CD} & \downarrow & \downarrow S_1 & \downarrow \\
 & & M'' & \xrightarrow{S_2} & \bullet \\
 & & \downarrow S_1 & \downarrow S_1 & \downarrow \\
 \bullet & \xrightarrow{S_2} & \bullet & \xrightarrow{S_2} & \bullet
 \end{array}$$

where the left rectangle can be closed by Lemma 35, and the top right and lower right squares can be closed by IH since we have that $\text{depth}(M') < \text{depth}(M)$ and $\text{depth}(M'') < \text{depth}(M)$. □

Lemma 37. $S_1 // S_2$

PROOF: By Lemma 36 and induction on the S_1 -derivation. □

4.13 General commutation and confluence

We have proved the 12 commutation lemmas of Table. 1, following the correspondence

- | | | |
|--------------------|--------------------|---------------------|
| (1) Cor. 6 p. 30 | (5) Lemma 12 p. 19 | (9) Lemma 30 p. 36 |
| (2) Lemma 10 p. 18 | (6) Lemma 13 p. 19 | (10) Lemma 17 p. 21 |
| (3) Cor. 7 p. 35 | (7) Cor. 5 p. 28 | (11) Lemma 34 p. 38 |
| (4) Lemma 11 p. 19 | (8) Lemma 26 p. 31 | (12) Lemma 37 p. 40 |

From this we can now deduce:

Proposition 4 (General commutation). — *Given two subsystems s_1 and s_2 , the following propositions are equivalent:*

1. $(s_1, s_2) \models \text{BCC}$ (binary closure conditions);
2. $s_1 \parallel_w s_2$ (weak commutation).
3. $s_1 \parallel s_2$ (weak commutation).

PROOF: From Prop. 2 we already now the equivalence 1. \Leftrightarrow 2.. Implication 2. \Rightarrow 3. follows from the divide and conquer technique combined with the proof of the 12 lemmas of Table. 1. Finally, 3. \Rightarrow 2. is obvious. \square

The proof of confluence for the entire 9-rule system is given in Fig. 5. (Notice that not all the 12 initial commutation lemmas are needed.)

Corollary 8. — $\lambda\mathcal{B}_C$ is conservative over $\lambda\eta$ -calculus, in the sense that:

$$\forall M_1, M_2 \in A \quad (\lambda\mathcal{B}_C \models M_1 = M_2 \Rightarrow \lambda\eta \models M_1 = M_2).$$

(Where A denotes the set of ordinary λ -terms.)

PROOF: Follows from Cor. 1 using the concluding remark of subsection 2.2. \square

5 Separation

5.1 Quasi-normal forms

Definition 8 (Head term). — *We call a head term (and write H, H_1, H' , etc.) any term that has one of the following four forms:*

$$\begin{array}{l} \mathbf{Head\ term} \quad H ::= x \quad | \quad c \\ \quad \quad \quad \quad | \{\theta\}.x \quad | \{\theta\}.c \quad (c \notin \text{dom}(\theta)) \end{array}$$

When a head term H is of one of the first three forms (variable, constructor, case binding on a variable), we say that H is defined. When H is of the last form (case binding on an unbound constructor), we say that H is undefined.

Definition 9 (Quasi-head normal form). — *A term M is said to be in quasi-head normal form if it has one of the following two forms*

$$\mathbf{Quasi-hnf} \quad M ::= \boxtimes \quad | \quad \lambda x_1 \cdots x_n. H N_1 \cdots N_k$$

where H is an arbitrary head term, called the head of M , and where N_1, \dots, N_k are arbitrary terms.

Each item of the following (mechanically constructed) proof states a commutation property $(s_1 // s_2)$ which is either:

- an item of Table 1;
- a consequence of $(s_1, s_2) \models \text{BCC}$ and $(s_1 + s_2) \models \text{SN}$;
- a consequence of two former items using the rule of inference:

$$\text{if } A // B \text{ and } A // C, \text{ then } A // (B + C).$$

1. $(\text{AL} \models \text{CR})$ [Table 1:1]
 2. $(\text{AL} // \text{AD})$ [Table 1:2]
 3. $(\text{AL} // \text{CO})$ [Table 1:4]
 4. $(\text{AL} // \text{CD})$ [Table 1:5]
 5. $(\text{AL} // \text{CL})$ [Table 1:6]
 6. $(\text{AL} // \text{CD} + \text{CL})$ since $(\text{CD} // \text{AL})$ [4.] and $(\text{CL} // \text{AL})$ [5.]
 7. $(\text{AL} // \text{AD} + \text{CD} + \text{CL})$ since $(\text{AD} // \text{AL})$ [2.] and $(\text{CD} + \text{CL} // \text{AL})$ [6.]
 8. $(\text{AL} // \text{AL} + \text{AD} + \text{CD} + \text{CL})$ since $(\text{AL} \models \text{CR})$ [1.] and $(\text{AD} + \text{CD} + \text{CL} // \text{AL})$ [7.]
 9. $(\text{AL} // \text{CC})$ [Table 1:7]
 10. $(\text{LA} + \text{LD} + \text{CD} + \text{CA} // \text{AL} + \text{AD} + \text{CD} + \text{CL})$ [Table 1:12]
 11. $(\text{AL} // \text{CL} + \text{CC})$ since $(\text{CL} // \text{AL})$ [5.] and $(\text{CC} // \text{AL})$ [9.]
 12. $(\text{AL} // \text{CD} + \text{CL} + \text{CC})$ since $(\text{CD} // \text{AL})$ [4.] and $(\text{CL} + \text{CC} // \text{AL})$ [11.]
 13. $(\text{AL} // \text{CO} + \text{CD} + \text{CL} + \text{CC})$ since $(\text{CO} // \text{AL})$ [3.] and $(\text{CD} + \text{CL} + \text{CC} // \text{AL})$ [12.]
 14. $(\text{AL} // \text{AD} + \text{CO} + \text{CD} + \text{CL} + \text{CC})$ since $(\text{AD} // \text{AL})$ [2.] and $(\text{CO} + \text{CD} + \text{CL} + \text{CC} // \text{AL})$ [13.]
 15. $(\text{AD} + \text{CD} + \text{CL} // \text{AD} + \text{CO} + \text{CD} + \text{CL} + \text{CC})$ since $\text{BCC} + \text{SN}$
 16. $(\text{AL} + \text{AD} + \text{CD} + \text{CL} // \text{AD} + \text{CO} + \text{CD} + \text{CL} + \text{CC})$ since $(\text{AL} // \text{AD} + \text{CO} + \text{CD} + \text{CL} + \text{CC})$ [14.] and $(\text{AD} + \text{CD} + \text{CL} // \text{AD} + \text{CO} + \text{CD} + \text{CL} + \text{CC})$ [15.]
 17. $(\text{AL} + \text{AD} + \text{CD} + \text{CL} // \text{AD} + \text{LA} + \text{LD} + \text{CO} + \text{CD} + \text{CA} + \text{CL} + \text{CC})$ since $(\text{LA} + \text{LD} + \text{CD} + \text{CA} // \text{AL} + \text{AD} + \text{CD} + \text{CL})$ [10.] and $(\text{AD} + \text{CO} + \text{CD} + \text{CL} + \text{CC} // \text{AL} + \text{AD} + \text{CD} + \text{CL})$ [16.]
 18. $(\text{AL} + \text{AD} + \text{CD} + \text{CL} // \text{AL} + \text{AD} + \text{LA} + \text{LD} + \text{CO} + \text{CD} + \text{CA} + \text{CL} + \text{CC})$ since $(\text{AL} // \text{AL} + \text{AD} + \text{CD} + \text{CL})$ [8.] and $(\text{AD} + \text{LA} + \text{LD} + \text{CO} + \text{CD} + \text{CA} + \text{CL} + \text{CC} // \text{AL} + \text{AD} + \text{CD} + \text{CL})$ [17.]
 19. $(\text{LA} + \text{LD} + \text{CO} + \text{CD} + \text{CA} + \text{CL} + \text{CC} // \text{AD} + \text{LA} + \text{LD} + \text{CO} + \text{CD} + \text{CA} + \text{CL} + \text{CC})$ since $\text{BCC} + \text{SN}$
 20. $(\text{AD} + \text{LA} + \text{LD} + \text{CO} + \text{CD} + \text{CA} + \text{CL} + \text{CC} // \text{AL} + \text{AD} + \text{LA} + \text{LD} + \text{CO} + \text{CD} + \text{CA} + \text{CL} + \text{CC})$ since $(\text{AL} + \text{AD} + \text{CD} + \text{CL} // \text{AD} + \text{LA} + \text{LD} + \text{CO} + \text{CD} + \text{CA} + \text{CL} + \text{CC})$ [17.] and $(\text{LA} + \text{LD} + \text{CO} + \text{CD} + \text{CA} + \text{CL} + \text{CC} // \text{AD} + \text{LA} + \text{LD} + \text{CO} + \text{CD} + \text{CA} + \text{CL} + \text{CC})$ [19.]
 21. $(\text{AL} + \text{AD} + \text{LA} + \text{LD} + \text{CO} + \text{CD} + \text{CA} + \text{CL} + \text{CC} \models \text{CR})$ since $(\text{AL} + \text{AD} + \text{CD} + \text{CL} // \text{AL} + \text{AD} + \text{LA} + \text{LD} + \text{CO} + \text{CD} + \text{CA} + \text{CL} + \text{CC})$ [18.] and $(\text{AD} + \text{LA} + \text{LD} + \text{CO} + \text{CD} + \text{CA} + \text{CL} + \text{CC} // \text{AL} + \text{AD} + \text{LA} + \text{LD} + \text{CO} + \text{CD} + \text{CA} + \text{CL} + \text{CC})$ [20.]
-

Fig. 5. Proof of confluence of $\lambda\mathcal{B}_c$

Here, the prefix ‘quasi’ expresses that such terms are in head normal form w.r.t. all the reduction rules, but (possibly) the rule LAMAPP (a.k.a. η). For instance, the term $\lambda x. cx$ is in quasi-head normal form according to the definition above, but still contains an η -redex at root position. In what follows, ‘quasi’ will systematically refer to ‘all the reduction rules but LAMAPP’.

As for head terms, we distinguish *defined* quasi-head normal forms from *undefined* ones, by saying that a quasi-head normal form M is *defined* when it has one of the forms

$$M ::= \lambda x_1 \cdots x_n. HN_1 \cdots N_k \quad (\text{where } H \text{ is defined})$$

and that M is *undefined* otherwise, that is, when M has the form

$$M ::= \lambda x_1 \cdots x_n. (\{\theta\}.c)N_1 \cdots N_k \quad (c \notin \text{dom}(\theta))$$

More generally, we call a *defined* term (resp. an *undefined* term) any term that reduces to a quasi-head normal form which is defined (resp. undefined). The class of defined terms is closed under arbitrary reduction, as for the class of undefined terms. Moreover, the class of undefined terms is closed under arbitrary substitution. (The notion of substitution will be defined in subsection 5.6.)

Definition 10 (Quasi-normal form). — *A term (resp. a case binding) is said to be in quasi-normal form when it is in normal form w.r.t. all the reduction rules but LAMAPP (a.k.a. η).*

Terms (resp. case bindings) that are in quasi-normal form are simply called *quasi-normal terms* (resp. *quasi-normal case bindings*). In particular, we call a *quasi-normal head term* any head term H which is in quasi-normal form. These notions have the following syntactic characterisation:

Proposition 5. *Quasi-normal terms, quasi-normal head terms, and quasi-normal case bindings are (mutually) characterised by the following BNF:*

$$\begin{array}{ll} \mathbf{Q.n.-terms} & N ::= \lambda x_1 \cdots x_n. HN_1 \cdots N_k \\ \mathbf{Q.n.-head-terms} & H ::= x \mid c \mid \{\theta\}.x \mid \{\theta\}.c \quad (c \notin \text{dom}(\theta)) \\ \mathbf{Q.n.-case bind.} & \theta ::= c_1 \mapsto N_1; \dots; c_p \mapsto N_p \end{array}$$

5.2 Separation contexts

The notion of *context with one hole* (notation $C[]$, $C'[]$, etc.) is defined in the λ -calculus with constructors as expected. The term obtained by filling a context with one hole $C[]$ with a term M is written $C[M]$, and the composition of two contexts $C[]$ and $C'[]$ is written $C'[C[]]$.

In what follows, we will mainly use contexts of a particular form, namely, *evaluation contexts*:

$$\mathbf{Evaluation contexts} \quad E[] ::= []N_1 \cdots N_n \mid \{\theta\}. []N_1 \cdots N_n$$

(The second form should be read $(\{\theta\}. \square)N_1 \cdots N_n$.)

Notice that the composition $E'[E[\square]]$ of two evaluation contexts $E[\square]$ and $E'[\square]$ is not always an evaluation context, but that it always reduces to an evaluation context using zero, one or several steps of the `CASEAPP` rule, possibly followed by a single step of the `CASECASE` rule:

$$\begin{aligned} & \left[\begin{array}{c} \square N_1 \cdots N_k \\ \{\theta\}. \square N_1 \cdots N_k \end{array} \right] N'_1 \cdots N'_{k'} = \square N_1 \cdots N_k N'_1 \cdots N'_{k'} \\ & \{\theta'\}. \left[\begin{array}{c} \square N_1 \cdots N_k \\ \{\theta\}. \square N_1 \cdots N_k \end{array} \right] N'_1 \cdots N'_{k'} \rightarrow^* \{\theta'\}. \square N_1 \cdots N_k N'_1 \cdots N'_{k'} \\ & \{\theta'\}. \left[\{\theta\}. \left[\begin{array}{c} \square N_1 \cdots N_k \\ \{\theta\}. \square N_1 \cdots N_k \end{array} \right] N'_1 \cdots N'_{k'} \right] \rightarrow^* \{\theta' \circ \theta\}. \square N_1 \cdots N_k N'_1 \cdots N'_{k'} \end{aligned}$$

Remark 3. An evaluation context $E[\square]$ can always be regarded as a term (of a particular form) that contains exactly one occurrence of a distinguished variable depicted \square —the hole. In particular, since the unique occurrence of the hole \square in an evaluation context $E[\square]$ is outside the scope of all the binders of $E[\square]$, the operation of replacement $E[M]$ works just as the ordinary operation of substitution $E\{\square := M\}$ of λ -calculus. (This is of course not the case for the general notion of context with one hole—think of $C[x]$ where $C[\square] = \lambda x. \square$.)

The daimon \blackboxtimes which represents immediate termination naturally absorbs all the evaluation contexts:

Lemma 38. *In any evaluation context $E[\square]$ one has $E[\blackboxtimes] \rightarrow^* \blackboxtimes$.*

Symmetrically, each subterm of the form $\{\theta\}.c$ (with $c \notin \text{dom}(\theta)$) blocks the computation process at head position so that undefined terms absorb all evaluation contexts as well:

Lemma 39. *Given an undefined term U , the term $E[U]$ is undefined in any evaluation context $E[\square]$.*

The daimon \blackboxtimes and undefined terms are thus natural candidates to define the notion of separation:

Definition 11 (Separability). *We say that two terms M_1 and M_2 are:*

- weakly separable *if there exists a context with one hole $C[\square]$ such that either:*
 - $C[M_1] \rightarrow^* \blackboxtimes$ and $C[M_2]$ is undefined, or
 - $C[M_2] \rightarrow^* \blackboxtimes$ and $C[M_1]$ is undefined;
- strongly separable *if there exists two contexts $C_1[\square]$ and $C_2[\square]$ such that*
 - $C_1[M_1] \rightarrow^* \blackboxtimes$ and $C_1[M_2]$ is undefined, and
 - $C_2[M_2] \rightarrow^* \blackboxtimes$ and $C_2[M_1]$ is undefined.

On the other hand, two undefined terms cannot be separated each other (precisely because their heads block all computations), so that we have to exclude them from our study of the separation property.

Definition 12 (Completely defined quasi-normal term). *A quasi-normal term M is said to be completely defined if it contains no subterm of the form $\{\theta\}.c$, where $c \notin \text{dom}(\theta)$.*

In what follows, we will show that distinct completely defined normal terms are weakly separable.

5.3 Tuples and Casuples

In order to retrieve subterms in a given term in normal form (the so called ‘Böhm-out’ technique), we need tuples, that are encoded as usual by setting $\langle M_1; \dots; M_n \rangle \equiv \lambda p. pM_1 \cdots M_n$. In what follows, we will use a slightly more general notation to represent the partial application of the n -uple constructor to its first k arguments and waiting the remaining $n - k$ arguments:

$$\langle M_1; \dots; M_k; *_{n-k} \rangle \equiv \lambda x_{k+1} \cdots x_n p. pM_1 \cdots M_k x_{k+1} \cdots x_n \quad (0 \leq k \leq n)$$

With these notations, the n -uple constructor is written $\langle *_{n-1} \rangle$. Its arguments are successively filled in as follows:

$$\langle *_{n-1} \rangle M_1 M_2 \cdots M_n \rightarrow \langle M_1; *_{n-1} \rangle M_2 \cdots M_n \rightarrow \cdots \rightarrow \langle M_1; M_2; \dots; M_n \rangle,$$

the $n + 1$ th argument finally acting as an eliminator:

$$\langle M_1; M_2; \dots; M_n \rangle P \rightarrow PM_1 M_2 \cdots M_n.$$

When a tuple—or a partial application of the n -uple constructor—is attacked on the lefthand side by a case construct

$$\{\theta\}. \langle \vec{M}; * \rangle \equiv \{\theta\}. \lambda \vec{x} p. p \vec{M} \vec{x} \rightarrow^* \lambda \vec{x} p. \{\theta\}. p \vec{M} \vec{x},$$

the case construct goes through all the abstractions and stops in front of the head variable (which is here the elimination variable). We then obtain an hybrid object formed as a combination of a case construct with a tuple—a *casuple*—that will be written:

$$\{\theta \mid M_1; \dots; M_k; *_{n-k}\} \equiv \lambda x_{k+1} \cdots x_n p. \{\theta\}. pM_1 \cdots M_k x_{k+1} \cdots x_n.$$

Casuples work just as ordinary tuples

$$\begin{aligned} \{\theta \mid *_{n-1}\} M_1 M_2 \cdots M_n &\rightarrow \{\theta \mid M_1; *_{n-1}\} M_2 \cdots M_n \\ &\rightarrow \cdots \rightarrow \{\theta \mid M_1; M_2; \dots; M_n\}, \end{aligned}$$

except that they preserve the entangled case construct through the successive partial applications. When the tuple is finally eliminated, the case is then restored and put in head position:

$$\{\theta \mid M_1; M_2; \dots; M_n\} P \rightarrow \{\theta\}. PM_1 M_2 \cdots M_n.$$

5.4 Encoding names

In the λ -calculus with constructors, booleans are represented by two reserved constructors **true** and **false**. The corresponding ‘if’ operator is then defined by

$$\text{if} \equiv \lambda bxy. \{\text{true} \mapsto x; \text{false} \mapsto y\}. b.$$

Natural numbers are represented using two reserved constructors 0 and s :

$$\bar{n} \equiv \underbrace{s(\dots(s0)\dots)}_n \quad (n \in \omega)$$

The function which tests equality between two natural numbers is defined as

$$\text{eq} \equiv \Theta(\lambda fxy. \{\!| 0 \mapsto \{\!| 0 \mapsto \text{true}; s \mapsto \lambda y'. \text{false} \}\!. y; \\ s \mapsto \lambda x'. \{\!| 0 \mapsto \text{false}; s \mapsto \lambda y'. f x' y' \}\!. y \}\!. x)$$

where $\Theta = (\lambda yf. f(yyf))(\lambda yf. f(yyf))$ is Turing's fixpoint combinator. We easily check that

$$\text{eq } \bar{n} \bar{m} \rightarrow^* \begin{cases} \text{true} & \text{if } n = m \\ \text{false} & \text{if } n \neq m \end{cases}$$

for all $n, m \in \omega$.

To each variable name x of the λ -calculus with constructors, we associate a numeral written \mathbf{x} (using the same name written in typewriter face), which we call the *symbol* of x . We assume that this encoding is into so that the term $\text{eq } \mathbf{x} \mathbf{y}$ reduces to true if and only if x and y are the same variable.

5.5 Disagreement

Definition 13 (Skeleton equivalence). *We say that two defined head terms H_1 and H_2 have the same skeleton and write $H_1 \approx H_2$ if either:*

- $H_1 = x_1$ and $H_2 = x_2$ for some $x_1, x_2 \in \mathcal{V}$, and $x_1 = x_2$; or
- $H_1 = c_1$ and $H_2 = c_2$ for some $c_1, c_2 \in \mathcal{C}$, and $c_1 = c_2$; or
- $H_1 = \{\!|\theta_1\}\!. x_1$ and $H_2 = \{\!|\theta_2\}\!. x_2$ for some case bindings θ_1, θ_2 and for some $x_1, x_2 \in \mathcal{V}$, and $\text{dom}(\theta_1) = \text{dom}(\theta_2)$ and $x_1 = x_2$.

Considering the negation of the former equivalence, it is clear that two defined head terms H_1 and H_2 have *not* the same skeleton ($H_1 \not\approx H_2$) when either:

- H_1 is a variable, and H_2 is a constructor (or symmetrically); or
- H_1 is a case-variable, and H_2 is a constructor (or symmetrically); or
- H_1 is a case-variable, and H_2 is a variable (or symmetrically); or
- H_1 and H_2 are both variables, but not the same variable; or
- H_1 and H_2 are both constructors, but not the same constructor; or
- $H_1 = \{\!|\theta_1\}\!. x_1$ and $H_2 = \{\!|\theta_2\}\!. x_2$ for some case bindings θ_1, θ_2 and for some $x_1, x_2 \in \mathcal{V}$, and either $x_1 \neq x_2$ or $\text{dom}(\theta_1) \neq \text{dom}(\theta_2)$.

(Notice that we do not consider the case of a head term of the form $\{\!|\theta\}\!. c$ where $c \notin \text{dom}(\theta)$, which is excluded from our definition.)

Definition 14 (Disagreement at depth d). *For each numeral $d \in \omega$, we define a binary relation on the class of completely defined quasi-normal terms, called the disagreement relation at depth d . This relation, written $\text{dis}_d(M_1, M_2)$ (M_1 and M_2 disagree at depth d), is defined by induction on $d \in \omega$ as follows:*

- (Base case) We write $\text{dis}_0(M_1, M_2)$ if either:
 - $M_1 = \mathbf{X}$ and $M_2 = \lambda x_1 \cdots x_n. H N_1 \cdots N_k$; or
 - $M_1 = \lambda x_1 \cdots x_n. H N_1 \cdots N_k$ and $M_2 = \mathbf{X}$; or
 - $M_1 = \lambda x_1 \cdots x_n. H_1 N_{1,1} \cdots N_{1,k_1}$ and $M_2 = \lambda x_1 \cdots x_n. H_2 N_{2,1} \cdots N_{2,k_2}$ and $H_1 \not\approx H_2$.
- (Inductive case) For all $d \in \omega$, we write $\text{dis}_{d+1}(M_1, M_2)$ if
 - $M_1 = \lambda x_1 \cdots x_n. H_1 N_{1,1} \cdots N_{1,k_1}$ and $M_2 = \lambda x_1 \cdots x_n. H_2 N_{2,1} \cdots N_{2,k_2}$ and $H_1 \approx H_2$, and if either
 - $H_1 = \{\theta_1\}.y$ and $H_2 = \{\theta_2\}.y$ for some case bindings θ_1, θ_2 and for some variable y , and there is a constructor $c \in \text{dom}(\theta_1) = \text{dom}(\theta_2)$ such that $\text{dis}_d(\theta_1(c), \theta_2(c))$; or
 - There is a position $1 \leq k \leq \min(k_1, k_2)$ such that $\text{dis}_d(N_{1,k}, N_{2,k})$.

Lemma 40 (Cooking lemma). *If M_1 and M_2 are two completely defined normal terms (w.r.t. all the reduction rules including $\text{LAMAPP} = \eta$) such that $M_1 \neq M_2$, then one can find two completely defined quasi-normal terms M'_1 and M'_2 such that $M'_1 \rightarrow_\eta^* M_1$, $M'_2 \rightarrow_\eta^* M_2$, and $\text{dis}_d(M'_1, M'_2)$ for some $d \in \omega$.*

PROOF: This is proved by induction on the maximum (or the sum) of the sizes of M_1 and M_2 , doing case analysis on M_1 and M_2 :

- Case 1. $M_1 \equiv \mathbf{X}$ and M_2 is of the form $M_2 \equiv \lambda x_1 \cdots x_n. H N_1 \cdots N_k$ (or symmetrically). Then M_1 and M_2 disagree at depth 0 by definition.
- Case 2. M_1 and M_2 are of the form
 - $M_1 \equiv \lambda x_1 \cdots x_{n_1}. H_1 N_{1,1} \cdots N_{1,k_1}$ and
 - $M_2 \equiv \lambda x_1 \cdots x_{n_2}. H_2 N_{2,1} \cdots N_{2,k_2}$.
 Without loss of generality, we can assume that $n_1 \leq n_2$, and that the first n_1 abstractions of M_2 use the same variable names as the n_1 abstractions of M_1 . We distinguish two cases, depending on $n_1 = n_2$ or $n_1 \neq n_2$.
 - Case 2.1. $n_1 = n_2 = n$. We distinguish three cases: $H_1 \not\approx H_2$, $H_1 \approx H_2$ but $H_1 \neq H_2$, and finally $H_1 = H_2$.
 - Case 2.1.1. $H_1 \not\approx H_2$. In this case, M_1 disagrees with M_2 at depth 0.
 - Case 2.1.2. $H_1 \approx H_2$ but $H_1 \neq H_2$. This means that H_1 and H_2 are of the form $H_1 = \{\theta_1\}.y$ and $H_2 = \{\theta_2\}.y$, with $\text{dom}(\theta_1) = \text{dom}(\theta_2)$. Since $H_1 \neq H_2$, there exists a constructor c and two terms P_1 and P_2 such that $(c \mapsto P_i) \in \theta_i$ for $i = 1, 2$, and $P_1 \neq P_2$. By induction hypothesis, there are defined quasi-normal terms $P'_i \rightarrow_\eta^* P_i$ (for $i = 1, 2$) such that $\text{dis}_d(P_1, P_2)$ for some $d \in \omega$. For $i = 1, 2$, let θ'_i be the case binding θ_i in which the binding $(c \mapsto P_i)$ has been replaced by the binding $(c \mapsto P'_i)$, and let $M'_i = \lambda x_1 \cdots x_n. (\{\theta'_i\}.y) N_{i,1} \cdots N_{i,k_i}$. We have $M'_i \rightarrow_\eta^* M_i$ for $i = 1, 2$, and $\text{dis}_{d+1}(M'_1, M'_2)$.
 - Case 2.1.3. $H_1 = H_2 = H$. In this case, our initial assumption $M_1 \neq M_2$ expresses that the lists $(N_{1,1}, \dots, N_{1,k_1})$ and $(N_{2,1}, \dots, N_{2,k_2})$ differ. Again, we distinguish two cases:

- Case 2.1.3.1. There exists $1 \leq k \leq \min(k_1, k_2)$ such that $N_{1,k} \neq N_{2,k}$. By induction hypothesis, there are terms $N'_{i,k}$ ($i = 1, 2$) such that $N'_{i,k} \rightarrow_{\eta}^* N_{i,k}$ and $\text{dis}_d(N'_{1,k}, N'_{2,k})$ for some $d \in \omega$. Let then define M'_i ($i = 1, 2$) from M_i by replacing the subterm $N_{i,k}$ by $N'_{i,k}$. We have $M'_i \rightarrow_{\eta}^* M_i$ for $i = 1, 2$, and $\text{dis}_{d+1}(M'_1, M'_2)$.
- Case 2.1.3.2. For all $1 \leq k \leq \min(k_1, k_2)$, one has $N_{1,k} = N_{2,k}$. From $M_1 \neq M_2$, we get $k_1 \neq k_2$. Without loss of generality, let us assume that $k_1 < k_2$, and consider a fresh variable y . Since $y \neq N_{2,k_1+1}$, there exists by induction hypothesis two terms N'_{1,k_1+1} and N'_{2,k_1+1} such that $N'_{1,k_1+1} \rightarrow_{\eta}^* y$, $N'_{2,k_1+1} \rightarrow_{\eta}^* N_{2,k_1+1}$ and $\text{dis}_d(N'_{1,k_1+1}, N'_{2,k_1+1})$ for some $d \in \omega$. Let us set $M'_1 = \lambda x_1 \cdots x_n y \cdot H N_{1,1} \cdots N_{1,k_1} N'_{1,k_1+1}$ and $M'_2 = \lambda x_1 \cdots x_n y \cdot H N_{2,1} \cdots N_{2,k_1} N'_{2,k_1+1} N_{2,k_1+2} \cdots N_{2,k_2} y$. By construction we have $M'_i \rightarrow_{\eta} M_i$ for $i = 1, 2$ and $\text{dis}_{d+1}(M'_1, M'_2)$.
- Case 2.2. $n_1 < n_2$. In order to keep the same number of abstractions in M_1 and M_2 , let us η -expand M_1 by letting $M'_1 = \lambda x_1 \cdots x_{n_2} \cdot H_1 N_{1,1} \cdots N_{1,k_1} x_{n_1+1} \cdots x_{n_2}$. Notice that $M'_1 \neq M_2$, since M_2 is in η -normal form whereas M'_1 is not. The rest of the proof proceeds as in case 2.1, replacing M_1 by M'_1 . (The reader is invited to check that the proof of case 2.1. only relies on the fact that $n_1 = n_2$, and does not use the fact that M_1 and M_2 are η -normal forms—which is no more the case when we replace M_1 by M'_1 .) □

5.6 Substitutions

A *substitution* is a finite set of pairs of the form

$$\sigma = \{(x_1 := M_1); \dots; (x_n := M_n)\}$$

(where x_1, \dots, x_n are variables and M_1, \dots, M_n terms). Given a term M (resp. a case binding θ) and a substitution σ , we write $M[\sigma]$ (resp. $\theta[\sigma]$) the term (resp. the case binding) defined by

$$\begin{aligned} x[\sigma] &\equiv \begin{cases} M & \text{if } x \text{ bound to } M \text{ in } \sigma \\ x & \text{if } x \text{ unbound in } \sigma \end{cases} \\ c[\sigma] &\equiv c \\ (\lambda x. M)[\sigma] &\equiv \lambda x. M[\sigma] && (x \text{ fresh w.r.t. } \sigma) \\ (MN)[\sigma] &\equiv M[\sigma]N[\sigma] \\ (\{\theta\}. M)[\sigma] &\equiv \{\theta[\sigma]\}. M[\sigma] \\ (c_1 \mapsto M_1; \dots; c_n \mapsto M_n)[\sigma] &\equiv (c_1 \mapsto M_1[\sigma]; \dots; c_n \mapsto M_n[\sigma]) \end{aligned}$$

Given a finite set of variables $X = \{x_1; \dots; x_n\}$ and an integer $K \geq 0$, we write

$$\sigma_X^K = \{(x_1 := \langle \mathbf{x}_1; *K \rangle); \dots; (x_n := \langle \mathbf{x}_n; *K \rangle)\}$$

the substitution which maps each variable $x_i \in X$ to a $(K + 1)$ uple constructor which is partially applied to the symbol \mathbf{x}_i of the variable x_i .

5.7 The separation theorem

Let M be a term in quasi-normal form. We call the *application strength* of M the largest integer $k \geq 0$ such that M has a subterm of the form $HN_1 \cdots N_k$.

Lemma 41. *Let M be a defined quasi-head normal term which is not the daimon, that is, a term of the form*

$$M \equiv \lambda x_1 \cdots x_n . HN_1 \cdots N_k$$

(where H is defined). Given a finite set X of variables such that $FV(M) \subset X$ and an integer $K \geq k$, one can find an evaluation context $E_1[]$ such that

$$E_1[M[\sigma_X^K]] \rightarrow^* \blacktriangleright$$

and another evaluation context $E_2[]$ such that

$$E_2[M[\sigma_X^K]] \text{ is undefined.}$$

PROOF: Let us write $X' = X \cup \{x_1; \dots; x_n\}$. We distinguish the following cases, depending on the shape of the head H .

1. $H \equiv y$ (variable). Take

$$E_1[] \equiv [] \langle \mathbf{x}_1; *K \rangle \cdots \langle \mathbf{x}_n; *K \rangle ?_{k+1} \cdots ?_K \blacktriangleright$$

where $?_{k+1}, \dots, ?_K$ are arbitrary terms. We have

$$\begin{aligned} E_1[M[\sigma_X^K]] &\equiv (\lambda x_1 \cdots x_n . y N_1 \cdots N_k)[\sigma_X^K] \langle \mathbf{x}_1; *K \rangle \cdots \langle \mathbf{x}_n; *K \rangle ?_{k+1} \cdots ?_K \blacktriangleright \\ &\rightarrow^* \langle y, *K \rangle N_1[\sigma_{X'}^K] \cdots N_k[\sigma_{X'}^K] ?_{k+1} \cdots ?_K \blacktriangleright \\ &\rightarrow^* \blacktriangleright y N_1[\sigma_{X'}^K] \cdots N_k[\sigma_{X'}^K] ?_{k+1} \cdots ?_K \\ &\rightarrow^* \blacktriangleright \end{aligned}$$

Similarly, for $E_2[]$ we take

$$E_2[] \equiv [] \langle \mathbf{x}_1; *K \rangle \cdots \langle \mathbf{x}_n; *K \rangle ?_{k+1} \cdots ?_K U$$

where $?_{k+1}, \dots, ?_K$ are arbitrary terms, and where U is an arbitrary undefined head term.

2. $H \equiv \{\theta\}.y$ (case variable). It suffices to take the very same evaluation contexts $E_1[]$ and $E_2[]$ as above. The additional case construct is absorbed both by the daimon \blacktriangleright and by the undefined head term U .
3. $H \equiv c$ (constructor). Set $E_1[] \equiv \{c \mapsto \blacktriangleright\}.[]$. We check that

$$\begin{aligned} E_1[M[\sigma_X^K]] &\equiv \{c \mapsto \blacktriangleright\}. (\lambda x_1 \cdots x_n . c N_1 \cdots N_k)[\sigma_X^K] \\ &\equiv (\{c \mapsto \blacktriangleright\}. (\lambda x_1 \cdots x_n . c N_1 \cdots N_k))[\sigma_X^K] \\ &\rightarrow^* (\lambda x_1 \cdots x_n . (\{c \mapsto \blacktriangleright\}. c) N_1 \cdots N_k)[\sigma_X^K] \\ &\rightarrow^* (\lambda x_1 \cdots x_n . \blacktriangleright N_1 \cdots N_k)[\sigma_X^K] \\ &\rightarrow^* \blacktriangleright \end{aligned}$$

Similarly, for $E_2[]$ we take $E_2[] \equiv \{\}.[]$ (empty case construct).

□

Proposition 6 (Separation of disagreeing terms). *Let $K \geq 0$ be a natural number, and M_1 and M_2 two completely defined quasi-normal terms whose application strength is less than or equal to K and such that M_1 and M_2 disagree at some depth $d \in \omega$. Then there exists an evaluation context $E[]$ such that either*

- $E[M_1[\sigma_X^K]] \rightarrow^* \mathfrak{X}$ and $E[M_2[\sigma_X^K]]$ is undefined, or
- $E[M_2[\sigma_X^K]] \rightarrow^* \mathfrak{X}$ and $E[M_1[\sigma_X^K]]$ is undefined, or

where X is any finite set of variables that contains at least the free variables of M_1 and M_2 , and where σ_X^K is the substitution defined in subsection 5.6.

PROOF: The proof proceeds by induction on the depth d of the disagreement between M_1 and M_2 . We distinguish the following cases:

1. M_1 and M_2 disagree at level 0. We distinguish the following cases:

- $M_1 = \mathfrak{X}$ and $M_2 = \lambda x_1 \cdots x_n . H N_1 \cdots N_k$.

In this case, take an evaluation context $E[]$ such that $E[M_2[\sigma_X^K]]$ is undefined (by Lemma 41) and conclude using Lemma 38.

- $M_1 = \lambda x_1 \cdots x_n . H N_1 \cdots N_k$ and $M_2 = \mathfrak{X}$.

This is the symmetric case (see above).

- $M_1 = \lambda x_1 \cdots x_n . H_1 N_{1,1} \cdots N_{1,k_1}$ and $M_2 = \lambda x_1 \cdots x_n . H_2 N_{2,1} \cdots N_{2,k_2}$

and $H_1 \not\approx H_2$. We distinguish the following cases, using the characterisation of the negation of skeleton equivalence (Def. 13) at the beginning of subsection 5.5:

- $H_1 \equiv y$ (where y is a variable), and $H_2 \equiv c$ (where c is a constructor).

Take:

$$E[] \equiv \{\{c \mapsto \mathfrak{X}\}. [] \langle x_1; *K \rangle \cdots \langle x_n; *K \rangle\}_{k_1+1} \cdots ?_K U$$

where $?_{k_1+1}, \dots, ?_K$ are arbitrary terms, and where U is an arbitrary undefined head term. Writing $X' = X \cup \{x_1; \dots; x_n\}$, we get

$$\begin{aligned} E[M_1[\sigma_X^K]] &\equiv (\{c \mapsto \mathfrak{X}\}. (\lambda x_1 \cdots x_n . y N_{1,1} \cdots N_{1,k_1})[\sigma_X^K]) \\ &\quad \langle x_1; *K \rangle \cdots \langle x_n; *K \rangle\}_{k_1+1} \cdots ?_K U \\ &\rightarrow^* (\lambda x_1 \cdots x_n . (\{c \mapsto \mathfrak{X}\}. y) N_{1,1} \cdots N_{1,k_1})[\sigma_X^K] \\ &\quad \langle x_1; *K \rangle \cdots \langle x_n; *K \rangle\}_{k_1+1} \cdots ?_K U \\ &\rightarrow^* (\{c \mapsto \mathfrak{X}\}. \langle y; *K \rangle) \\ &\quad N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K]\}_{k_1+1} \cdots ?_K U \\ &\rightarrow^* \{\{c \mapsto \mathfrak{X} \mid y; *K\} N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K]\}_{k_1+1} \cdots ?_K U \\ &\rightarrow^* (\{c \mapsto \mathfrak{X}\}. U) y N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K]\}_{k_1+1} \cdots ?_K, \end{aligned}$$

the latter term being undefined by lemma 39, whereas

$$\begin{aligned} E[M_2[\sigma_X^K]] &\equiv (\{c \mapsto \mathfrak{X}\}. (\lambda x_1 \cdots x_n . c N_{2,1} \cdots N_{2,k_2})[\sigma_X^K]) \\ &\quad \langle x_1; *K \rangle \cdots \langle x_n; *K \rangle\}_{k_1+1} \cdots ?_K U \\ &\rightarrow^* (\lambda x_1 \cdots x_n . (\{c \mapsto \mathfrak{X}\}. c) N_{2,1} \cdots N_{2,k_2})[\sigma_X^K] \\ &\quad \langle x_1; *K \rangle \cdots \langle x_n; *K \rangle\}_{k_1+1} \cdots ?_K U \\ &\rightarrow^* (\lambda x_1 \cdots x_n . \mathfrak{X} N_{2,1} \cdots N_{2,k_2})[\sigma_X^K] \\ &\quad \langle x_1; *K \rangle \cdots \langle x_n; *K \rangle\}_{k_1+1} \cdots ?_K U \\ &\rightarrow^* \mathfrak{X} \end{aligned}$$

The symmetric case is treated the same way.

- $H_1 \equiv \{\theta\}.y$ (where θ is a case binding and where y is a variable), and $H_2 \equiv c$ (where c is a constructor). Again, let us take

$$E[] \equiv \{\{c \mapsto \mathfrak{X}\}\}. [] \langle \mathbf{x}_1; *K \rangle \cdots \langle \mathbf{x}_n; *K \rangle ?_{k_1+1} \cdots ?_K U$$

where $?_{k_1+1}, \dots, ?_K$ are arbitrary terms, and where U is an arbitrary undefined head term. As in the latter case, we check that $E[M_1[\sigma_X^K]]$ is undefined whereas $E[M_2[\sigma_X^K]] \rightarrow^* \mathfrak{X}$. (The reader is invited to check that the additional case $\{\theta\}._$ plays no essential role during reduction.) The symmetric case is treated the same way.

- $H_1 \equiv \{\theta\}.y_1$ (where θ is a case binding and where y_1 is a variable), and $H_2 \equiv y_2$ (where y_2 is a variable). Take

$$E[] \equiv \{\{c \mapsto \mathfrak{X}\}\}. [] \langle \mathbf{x}_1; *K \rangle \cdots \langle \mathbf{x}_n; *K \rangle \underbrace{c \cdots c}_{K+1 \text{ times}}$$

where c is an arbitrary constructor such that $c \notin \text{dom}(\theta)$. Writing $X' = X \cup \{x_1; \dots; x_n\}$, we get

$$\begin{aligned} E[M_1[\sigma_X^K]] &\equiv (\{\{c \mapsto \mathfrak{X}\}\}. (\lambda x_1 \cdots x_n. \{\theta\}. y_1 N_{1,1} \cdots N_{1,k_1})[\sigma_X^K]) \\ &\quad \langle \mathbf{x}_1; *K \rangle \cdots \langle \mathbf{x}_n; *K \rangle \underbrace{c \cdots c}_{K+1} \\ &\rightarrow^* (\lambda x_1 \cdots x_n. (\{\{c \mapsto \mathfrak{X}\}\} \circ \theta). y_1) N_{1,1} \cdots N_{1,k_1}[\sigma_X^K] \\ &\quad \langle \mathbf{x}_1; *K \rangle \cdots \langle \mathbf{x}_n; *K \rangle \underbrace{c \cdots c}_{K+1} \\ &\rightarrow^* (\{\{c \mapsto \mathfrak{X}\}\} \circ \theta[\sigma_{X'}^K]). \langle \mathbf{y}_1; *K \rangle \\ &\quad N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] \underbrace{c \cdots c}_{K+1} \\ &\rightarrow^* \{\{c \mapsto \mathfrak{X}\}\} \circ \theta[\sigma_{X'}^K] \mid \mathbf{y}_1; *K \\ &\quad N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] \underbrace{c \cdots c}_{K+1} \\ &\rightarrow^* (\{\{c \mapsto \mathfrak{X}\}\} \circ \theta[\sigma_{X'}^K]). c \\ &\quad y_1 N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] \underbrace{c \cdots c}_K \end{aligned}$$

this term being undefined since $c \notin \text{dom}((c \mapsto \mathfrak{X}) \circ \theta) = \text{dom}(\theta)$. On the other hand we have

$$\begin{aligned} E[M_2[\sigma_X^K]] &\rightarrow^* (\{\{c \mapsto \mathfrak{X}\}\}. c) y_2 N_{2,1}[\sigma_{X'}^K] \cdots N_{2,k_2}[\sigma_{X'}^K] \underbrace{c \cdots c}_K \\ &\rightarrow^* \mathfrak{X} y_2 N_{2,1}[\sigma_{X'}^K] \cdots N_{2,k_2}[\sigma_{X'}^K] \underbrace{c \cdots c}_K \\ &\rightarrow^* \mathfrak{X} \end{aligned}$$

The symmetric case is treated the same way.

- $H_1 \equiv y_1$ and $H_2 \equiv y_2$ (where y_1 and y_2 are variables), but $y_1 \neq y_2$. Take

$$\begin{aligned} P &\equiv \lambda z. \text{if } (\text{eq } z \ y_1) \ \mathfrak{X} \ U \\ E[] &\equiv [] \langle \mathbf{x}_1; *K \rangle \cdots \langle \mathbf{x}_n; *K \rangle \underbrace{P \cdots P}_{K+1 \text{ times}} \end{aligned}$$

where U is an arbitrary undefined head term. Writing $X' = X \cup \{x_1; \dots; x_n\}$, we get

$$\begin{aligned}
E[M_1[\sigma_X^K]] &\equiv (\lambda x_1 \dots x_n . y_1 N_{1,1} \dots N_{1,k_1})[\sigma_X^K] \\
&\quad \langle \mathbf{x}_1; *K \rangle \dots \langle \mathbf{x}_n; *K \rangle \underbrace{P \dots P}_{K+1} \\
&\rightarrow^* \langle y_1; *K \rangle N_{1,1}[\sigma_{X'}^K] \dots N_{1,k_1}[\sigma_{X'}^K] \underbrace{P \dots P}_{K+1} \\
&\rightarrow^* P y_1 N_{1,1}[\sigma_{X'}^K] \dots N_{1,k_1}[\sigma_{X'}^K] \underbrace{P \dots P}_{K+1} \\
&\rightarrow^* \boxtimes N_{1,1}[\sigma_{X'}^K] \dots N_{1,k_1}[\sigma_{X'}^K] \underbrace{P \dots P}_K \\
&\rightarrow^* \boxtimes
\end{aligned}$$

On the other hand we have

$$\begin{aligned}
E[M_2[\sigma_X^K]] &\rightarrow^* P y_2 N_{2,1}[\sigma_{X'}^K] \dots N_{2,k_2}[\sigma_{X'}^K] \underbrace{P \dots P}_K \\
&\rightarrow^* U N_{2,1}[\sigma_{X'}^K] \dots N_{2,k_2}[\sigma_{X'}^K] \underbrace{P \dots P}_K
\end{aligned}$$

which is undefined by lemma 39.

- $H_1 \equiv c_1$ and $H_2 \equiv c_2$ (where c_1 and c_2 are constructors), but $c_1 \not\equiv c_2$.
Take

$$\begin{aligned}
\theta &\equiv (c_1 \mapsto \boxtimes; c_2 \mapsto U) \\
E[] &\equiv \{\theta\}. []
\end{aligned}$$

where U is an arbitrary undefined closed head term. We check that

$$\begin{aligned}
E[M_1[\sigma_X^K]] &\equiv \{\theta\}. (\lambda x_1 \dots x_n . c_1 N_{1,1} \dots N_{1,k_1})[\sigma_X^K] \\
&\rightarrow^* (\lambda x_1 \dots x_n . (\{\theta\}. c_1) N_{1,1} \dots N_{1,k_1})[\sigma_X^K] \\
&\rightarrow^* (\lambda x_1 \dots x_n . \boxtimes N_{1,1} \dots N_{1,k_1})[\sigma_X^K] \\
&\rightarrow^* \boxtimes
\end{aligned}$$

whereas

$$\begin{aligned}
E[M_2[\sigma_X^K]] &\rightarrow^* (\lambda x_1 \dots x_n . (\{\theta\}. c_2) N_{2,1} \dots N_{2,k_2})[\sigma_X^K] \\
&\rightarrow^* (\lambda x_1 \dots x_n . U N_{2,1} \dots N_{2,k_2})[\sigma_X^K]
\end{aligned}$$

which is undefined.

- $H_1 = \{\theta_1\}. y_1$ and $H_2 = \{\theta_2\}. y_2$ for some case bindings θ_1, θ_2 and for some variables y_1, y_2 , and $y_1 \not\equiv y_2$. This case is treated similarly to the case where H_1 and H_2 are distinct variables, by setting

$$\begin{aligned}
P &\equiv \lambda z . \text{if } (\text{eq } z y_1) \boxtimes U \\
E[] &\equiv [] \langle \mathbf{x}_1; *K \rangle \dots \langle \mathbf{x}_n; *K \rangle \underbrace{P \dots P}_{K+1 \text{ times}}
\end{aligned}$$

(where U is an arbitrary undefined head term). The reader is invited to check that the presence of additional case constructs $\{\theta_i\}. _$ does not essentially change how reduction proceeds.

- $H_1 = \{\theta_1\}.y$ and $H_2 = \{\theta_2\}.y$ for some case bindings θ_1, θ_2 and for some variable y , and $\text{dom}(\theta_1) \neq \text{dom}(\theta_2)$. Without loss of generality, assume that c_1 is a constructor such that $c_1 \in \text{dom}(\theta_1)$ and $c_1 \notin \text{dom}(\theta_2)$ (the other case is treated by symmetry). Writing $X' = X \cup \{x_1; \dots; x_n\}$ as usual, let us set:

$$\begin{aligned} P &\equiv \lambda z_0 z_1 \dots z_{2K} . c_1 \\ E_0[] &\equiv [] \langle \mathbf{x}_1; *K \rangle \dots \langle \mathbf{x}_n; *K \rangle \underbrace{P \dots P}_{K+1} \end{aligned}$$

We have:

$$\begin{aligned} E_0[M_1[\sigma_X^K]] &\equiv (\lambda x_1 \dots x_n . \{\theta_1\}.y N_{1,1} \dots N_{1,k_1})[\sigma_X^K] \\ &\quad \langle \mathbf{x}_1; *K \rangle \dots \langle \mathbf{x}_n; *K \rangle \underbrace{P \dots P}_{K+1} \\ &\rightarrow^* (\{\theta_1[\sigma_{X'}^K]\}. \langle y; *K \rangle) N_{1,1}[\sigma_{X'}^K] \dots N_{1,k_1}[\sigma_{X'}^K] \underbrace{P \dots P}_{K+1} \\ &\rightarrow^* \{\theta_1[\sigma_{X'}^K] \mid y; *K\} N_{1,1}[\sigma_{X'}^K] \dots N_{1,k_1}[\sigma_{X'}^K] \underbrace{P \dots P}_{K+1} \\ &\rightarrow^* (\{\theta_1[\sigma_{X'}^K]\}. P) y N_{1,1}[\sigma_{X'}^K] \dots N_{1,k_1}[\sigma_{X'}^K] \underbrace{P \dots P}_{K+1} \\ &\rightarrow^* (\lambda z_0 z_1 \dots z_{2K} . \{\theta_1[\sigma_{X'}^K]\}. c_1) \\ &\quad y N_{1,1}[\sigma_{X'}^K] \dots N_{1,k_1}[\sigma_{X'}^K] \underbrace{P \dots P}_K \\ &\rightarrow^* \lambda z_{k_1+K+1} \dots z_{2K} . \{\theta_1[\sigma_{X'}^K]\}. c_1 \\ &\rightarrow^* \lambda z_{k_1+K+1} \dots z_{2K} . \theta_1(c_1)[\sigma_{X'}^K] \end{aligned}$$

whereas

$$E_0[M_2[\sigma_X^K]] \rightarrow^* \lambda z_{k_2+K+1} \dots z_{2K} . \{\theta_2[\sigma_{X'}^K]\}. c_1$$

which is undefined. The term $\theta_1(c_1)$, which is a subterm of M_1 , is a completely defined quasi-normal form, and so is the term

$$T \equiv \lambda z_{k_1+K+1} \dots z_{2K} . \theta_1(c_1).$$

Thus by lemma 41, there exists an evaluation context $F[]$ such that

$$F[T[\sigma_{X'}^K]] \equiv F[\lambda z_{k_1+K+1} \dots z_{2K} . \theta_1(c_1)[\sigma_{X'}^K]] \rightarrow^* \boxtimes$$

Finally, let us set $E[] = F[E_0[]]$. We then get

$$E[M_1[\sigma_{X'}^K]] \equiv F[E_0[M_1[\sigma_{X'}^K]]] \rightarrow^* F[T[\sigma_{X'}^K]] \rightarrow^* \boxtimes$$

whereas

$$E[M_2[\sigma_{X'}^K]] \rightarrow^* F[\lambda z_{k_2+K+1} \dots z_{2K} . \{\theta_2[\sigma_{X'}^K]\}. c_1]$$

which is undefined by lemma 39.

2. M_1 and M_2 disagree at level $d+1$. By definition, M_1 and M_2 are of the form $M_1 = \lambda x_1 \cdots x_n . H_1 N_{1,1} \cdots N_{1,k_1}$ and $M_2 = \lambda x_1 \cdots x_n . H_2 N_{2,1} \cdots N_{2,k_2}$, with $H_1 \approx H_2$. Let us write $X' = X \cup \{x_1; \dots; x_n\}$. By definition of the relation of disagreement (at depth $d+1$), there are two possible cases:
- There is a position $1 \leq k \leq \min(k_1, k_2)$ such that $\text{dis}_d(N_{1,k}, N_{2,k})$. By induction hypothesis, there exists an evaluation context $E_0[]$ such that

$$E_0[N_{1,k}[\sigma_{X'}^K]] \rightarrow^* \blacktriangleright \quad \text{and} \quad E_0[N_{2,k}[\sigma_{X'}^K]] \text{ is undefined}$$

(or conversely). By case distinction on the shape of $H_1 \approx H_2$:

- $H_1 \equiv H_2 \equiv y$. Let us set

$$\begin{aligned} P &\equiv \lambda z_0 z_1 \dots z_K . E_0[z_k] \\ E[] &\equiv [] \langle \mathbf{x}_1; *K \rangle \cdots \langle \mathbf{x}_n; *K \rangle \underbrace{P \cdots P}_{K+1} \end{aligned}$$

We check that

$$\begin{aligned} E[M_1[\sigma_X^K]] &\equiv (\lambda x_1 \dots x_n . y N_{1,1} \cdots N_{1,k_1})[\sigma_X^K] \\ &\quad \langle \mathbf{x}_1; *K \rangle \cdots \langle \mathbf{x}_n; *K \rangle \underbrace{P \cdots P}_{K+1} \\ &\rightarrow^* \langle y; *K \rangle N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] \underbrace{P \cdots P}_{K+1} \\ &\rightarrow^* P y N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] \underbrace{P \cdots P}_K \\ &\rightarrow^* E_0[N_{1,k}[\sigma_{X'}^K]] \underbrace{P \cdots P}_{k_1} \\ &\rightarrow^* \blacktriangleright \underbrace{P \cdots P}_{k_1} \\ &\rightarrow^* \blacktriangleright \end{aligned}$$

whereas

$$E[M_2[\sigma_X^K]] \rightarrow^* E_0[N_{2,k}[\sigma_{X'}^K]] \underbrace{P \cdots P}_{k_2}$$

which is undefined by lemma 39.

- $H_1 \equiv \{\theta_1\}. y$ and $H_2 \equiv \{\theta_2\}. y$. This case is treated similarly to the latter case, using the same evaluation context $E[]$.
- $H_1 \equiv H_2 \equiv c$. Let us set

$$\begin{aligned} \theta &\equiv (c \mapsto \lambda z_1 \dots z_k . E_0[z_k]) \\ E[] &\equiv \{\theta\}. [] \langle \mathbf{x}_1; *K \rangle \cdots \langle \mathbf{x}_n; *K \rangle \end{aligned}$$

We check that

$$\begin{aligned} E[M_1[\sigma_X^K]] &\equiv \{\theta\}. (\lambda x_1 \dots x_n . c N_{1,1} \cdots N_{1,k_1})[\sigma_X^K] \\ &\quad \langle \mathbf{x}_1; *K \rangle \cdots \langle \mathbf{x}_n; *K \rangle \\ &\rightarrow^* (\lambda x_1 \dots x_n . (\{\theta\}. c) N_{1,1} \cdots N_{1,k_1})[\sigma_X^K] \\ &\quad \langle \mathbf{x}_1; *K \rangle \cdots \langle \mathbf{x}_n; *K \rangle \\ &\rightarrow^* (\{\theta\}. c) N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] \\ &\rightarrow^* E_0[N_{1,k}[\sigma_{X'}^K]] N_{1,k+1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] \\ &\rightarrow^* \blacktriangleright N_{1,k+1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] \\ &\rightarrow^* \blacktriangleright \end{aligned}$$

whereas

$$\begin{aligned} E[M_2[\sigma_X^K]] &\rightarrow^* (\{\theta\}.c)N_{2,1}[\sigma_{X'}^K] \cdots N_{2,k_2}[\sigma_{X'}^K] \\ &\rightarrow^* E_0[N_{2,k}[\sigma_{X'}^K]]N_{2,k+1}[\sigma_{X'}^K] \cdots N_{2,k_2}[\sigma_{X'}^K] \end{aligned}$$

which is undefined by lemma 39.

- $H_1 = \{\theta_1\}.y$ and $H_2 = \{\theta_2\}.y$ for some case bindings θ_1, θ_2 and for some variable y , and there is a constructor $c \in \text{dom}(\theta_1) = \text{dom}(\theta_2)$ such that $\text{dis}_d(\theta_1(c), \theta_2(c))$. Again, we distinguish two cases, depending on whether $k_1 = k_2$ or not.

- (a) $k_1 = k_2 = k$. By induction hypothesis, we know that there exists an evaluation context $E_0[]$ such that

$$E_0[\theta_1(c)[\sigma_{X'}^K]] \rightarrow^* \boxtimes \quad \text{and} \quad E_0[\theta_2(c)[\sigma_{X'}^K]] \text{ is undefined}$$

(or conversely). Let us then set

$$\begin{aligned} P &\equiv \lambda z_0 z_1 \dots z_K . c \\ E'[] &\equiv [] \langle \mathbf{x}_1; *K \rangle \cdots \langle \mathbf{x}_n; *K \rangle \underbrace{P \cdots P}_{K+1-k} \end{aligned}$$

We have

$$\begin{aligned} E'[M_1[\sigma_X^K]] &\equiv (\lambda x_1 \cdots x_n . \{\theta_1\}.y N_{1,1} \cdots N_{1,k})[\sigma_X^K] \\ &\quad \langle \mathbf{x}_1; *K \rangle \cdots \langle \mathbf{x}_n; *K \rangle \underbrace{P \cdots P}_{K+1-k} \\ &\rightarrow^* (\{\theta_1[\sigma_{X'}^K]\}. \langle \mathbf{y}, *K \rangle) N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k}[\sigma_{X'}^K] \underbrace{P \cdots P}_{K+1-k} \\ &\rightarrow^* \{\theta_1[\sigma_{X'}^K] \mid \mathbf{y}, *K\} N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k}[\sigma_{X'}^K] \underbrace{P \cdots P}_{K+1-k} \\ &\rightarrow^* (\{\theta_1[\sigma_{X'}^K]\}. P) y N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k}[\sigma_{X'}^K] \underbrace{P \cdots P}_{K-k} \\ &\rightarrow^* (\lambda z_0 z_1 \dots z_K . \{\theta_1[\sigma_{X'}^K]\}. c) \\ &\quad y N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k}[\sigma_{X'}^K] \underbrace{P \cdots P}_{K-k} \\ &\rightarrow^* \{\theta_1[\sigma_{X'}^K]\}. c \rightarrow \theta_1(c)[\sigma_{X'}^K] \end{aligned}$$

Similarly we have

$$E'[M_2[\sigma_X^K]] \rightarrow^* \{\theta_2[\sigma_{X'}^K]\}.c \rightarrow \theta_2(c)[\sigma_{X'}^K].$$

Thus if we take $E[] \equiv E_0[E'[]]$ we get

$$E[M_1[\sigma_X^K]] \equiv E_0[E'[M_1[\sigma_X^K]]] \rightarrow^* E_0[\theta_1(c)[\sigma_{X'}^K]] \rightarrow^* \boxtimes$$

whereas

$$E[M_2[\sigma_X^K]] \equiv E_0[E'[M_2[\sigma_X^K]]] \rightarrow^* E_0[\theta_2(c)[\sigma_{X'}^K]],$$

which is undefined.

(b) $k_1 \neq k_2$. Without loss of generality, assume $k_1 < k_2$ and set

$$E[] \equiv [] \langle \mathbf{x}_1; *K \rangle \cdots \langle \mathbf{x}_n; *K \rangle \underbrace{? \cdots ?}_{K-k_2} U \underbrace{? \cdots ?}_{k_2-k_1-1} \boxtimes$$

where ‘?’s denote arbitrary terms, and where U is any undefined head term. We then get

$$\begin{aligned} E[M_1[\sigma_X^K]] &\equiv (\lambda x_1 \cdots x_n . \{\theta_1\} . y N_{1,1} \cdots N_{1,k_1})[\sigma_X^K] \\ &\quad \langle \mathbf{x}_1; *K \rangle \cdots \langle \mathbf{x}_n; *K \rangle \underbrace{? \cdots ?}_{K-k_2} U \underbrace{? \cdots ?}_{k_2-k_1-1} \boxtimes \\ &\rightarrow^* (\{\theta_1[\sigma_X^K]\} . \langle y; *K \rangle) \\ &\quad N_{1,1}[\sigma_X^K] \cdots N_{1,k_1}[\sigma_X^K] \underbrace{? \cdots ?}_{K-k_2} U \underbrace{? \cdots ?}_{k_2-k_1-1} \boxtimes \\ &\rightarrow^* \{\theta_1[\sigma_X^K] \mid y; *K\} \\ &\quad N_{1,1}[\sigma_X^K] \cdots N_{1,k_1}[\sigma_X^K] \underbrace{? \cdots ?}_{K-k_2} U \underbrace{? \cdots ?}_{k_2-k_1-1} \boxtimes \\ &\rightarrow^* \{\theta_1[\sigma_X^K]\} . \boxtimes y N_{1,1}[\sigma_X^K] \cdots N_{1,k_1}[\sigma_X^K] \underbrace{? \cdots ?}_{K-k_2} U \underbrace{? \cdots ?}_{k_2-k_1-1} \\ &\rightarrow^* \boxtimes \end{aligned}$$

whereas

$$\begin{aligned} E[M_2[\sigma_X^K]] &\rightarrow^* \{\theta_2[\sigma_X^K] \mid y; *K\} \\ &\quad N_{2,1}[\sigma_X^K] \cdots N_{2,k_2}[\sigma_X^K] \underbrace{? \cdots ?}_{K-k_2} U \underbrace{? \cdots ?}_{k_2-k_1-1} \boxtimes \\ &\rightarrow^* \{\theta_2[\sigma_X^K]\} . U y N_{2,1}[\sigma_X^K] \cdots N_{2,k_2}[\sigma_X^K] \underbrace{? \cdots ?}_{K-k_2} \underbrace{? \cdots ?}_{k_2-k_1-1} \boxtimes \end{aligned}$$

which is undefined. \square

Theorem 2 (Separation). *Let M_1 and M_2 be completely defined terms in normal form. If $M_1 \not\equiv M_2$, then M_1 and M_2 are weakly separable.*

PROOF: Assume M_1 and M_2 are distinct normal terms. By the cooking lemma, there exists two completely defined quasi-normal terms M'_1 and M'_2 such that $M'_i \rightarrow_\eta^* M_i$ (for $i = 1, 2$) and $\text{dis}_d(M'_1, M'_2)$ for some $d \in \omega$. Set $X = FV(M'_1) \cup FV(M'_2)$ and define K as the maximum of the application strengths of M'_1 and M'_2 . From the latter proposition, there exists an evaluation context $E[]$ such that

$$E[M'_1[\sigma_X^K]] \rightarrow^* \boxtimes \quad \text{and} \quad E[M'_2[\sigma_X^K]] \text{ undefined} \quad (\text{or symmetrically})$$

It suffices to set

$$C[] \equiv E[(\lambda x_1 \cdots x_n . [])\sigma_X^K(x_1) \cdots \sigma_X^K(x_n)]$$

(where x_1, \dots, x_n are the elements of X) and we are done using the Church-Rosser property. \square

6 Conclusion

We have introduced an extension of λ -calculus, $\lambda\mathcal{B}_C$, in which pattern matching is implemented via a mechanism of case analysis that behaves like a head linear substitution over constructors. We have shown that the reduction relation of $\lambda\mathcal{B}_C$ is confluent and conservative over the $\lambda\eta$ -calculus, but also that it is complete in the sense that it provides sufficiently many reduction rules to identify all observationally equivalent normalisable terms.

Using the divide-and-conquer method for other proofs of confluence An original aspect of this work is the way we proved confluence by systematically studying the commutation properties of all pairs of subsystems of $\lambda\mathcal{B}_C$. Surprisingly, the mechanical propagation rule

$$\text{“if } A // B \text{ and } A // C \text{ then } A // (B + C)\text{”}$$

(combined with the primitive knowledge of all commutation properties between subsystems that do not involve `APPLAM`) is sufficient to reduce the proof of the expected 7,784 non-trivial commutation lemmas to only 12 primitive lemmas, that are established by hand. It would be interesting to investigate further to see whether the same method can be used to prove the confluence of other rewrite systems with many reduction rules—typically, systems with explicit substitutions.

A notion of Böhm tree for $\lambda\mathcal{B}_C$ The separation theorem we proved suggests that head normal forms of $\lambda\mathcal{B}_C$ could be the adequate brick to define a notion of Böhm-tree [3, 2] for $\lambda\mathcal{B}_C$ —and more generally, for ML-style pattern-matching. However, the fact that it is a *weak* separation theorem also suggests that the observational ordering is non-trivial on the set of normal forms. Characterising observational ordering on normal forms could be the next step to deepen our understanding of both operational and denotational semantics of $\lambda\mathcal{B}_C$.

Which type system for $\lambda\mathcal{B}_C$? The reduction rules `CASEAPP` and `CASELAM` which are the starting point of this work deeply challenge the traditional intuition of the notion of type, for which functions and constructed values live in different worlds. However, the good operational semantics of the calculus naturally raises the exciting question of finding a suitable type system for $\lambda\mathcal{B}_C$.

References

1. F. Baader and T. Nipkow. *Rewriting and All That*. Addison-Wesley, 1999.
2. H. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and The Foundations of Mathematics*. North-Holland, 1984.
3. C. Böhm, M. Dezani-Ciancaglini, P. Peretti, and S. Ronchi Della Rocha. A discrimination algorithm inside lambda-beta-calculus. *Theoretical Computer Science*, 8(3):265–291, 1979.

4. S. Cerrito and D. Kesner. Pattern matching as cut elimination. In *Logics In Computer Science (LICS'99)*, pages 98–108, 1999.
5. A. Church. *The calculi of lambda-conversion*, volume 6 of *Annals of Mathematical Studies*. Princeton, 1941.
6. H. Cirstea and C. Kirchner. Rho-calculus, the rewriting calculus. In *5th International Workshop on Constraints in Computational Logics*, 1998.
7. J.-Y. Girard. Locus solum: From the rules of logic to the logic of rules. *Mathematical Structures in Computer Science*, 11(3):301–506, 2001.
8. P. Hudak, S. Peyton-Jones, and P. Wadler. Report on the programming language Haskell, a non-strict, purely functional language (Version 1.2). *Sigplan Notices*, 1992.
9. C. Barry Jay. The pattern calculus. *ACM Transactions on Programming Languages and Systems*, 26(6):911–937, 2004.
10. W. Kahl. Basic pattern matching calculi: Syntax, reduction, confluence, and normalisation. Technical Report 16, Software Quality Research Laboratory, McMaster Univ., 2003.
11. R. Milner, M. Tofte, and R. Harper. *The definition of Standard ML*. MIT Press, 1990.
12. The Objective Caml language. <http://caml.inria.fr/>.
13. A. Ríos. *Contribution à l'étude des λ -calculs avec substitutions explicites*. PhD thesis, Université Paris 7, 1993.
14. V. van Oostrom. Lambda calculus with patterns. Technical Report IR-228, Vrije Universiteit, Amsterdam, 1990.