JFP **19** (5): 581–631, 2009. © Cambridge University Press 2009
doi:10.1017/S0956796809007369

The λ-calculus with constructors: Syntax, confluence and separation

ARIEL ARBISER
Departamento de Computación – Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Argentina
(*e-mail:* arbiser@dc.uba.ar)

ALEXANDRE MIQUEL
PPS & Université Paris 7 – Case 7014, 2 Place Jussieu, 75251 PARIS Cedex 05, France
(*e-mail:* alexandre.miquel@pps.jussieu.fr)

ALEJANDRO RÍOS
Departamento de Computación – Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Argentina
(*e-mail:* rios@dc.uba.ar)

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 1930s (Church 1941) 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 (Barendregt 1984), such as Church–Rosser 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 (Milner *et al.* 1990; Hudak *et al.* 1992; The Objective Caml language) and in rewriting (van Oostrom 1990; Cirstea & Kirchner 1998; Cerrito & Kesner 1999; Kahl 2003; Jay 2004; Jay & Kesner 2006).

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 (Böhm *et al.* 1979) 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-typed[1]) reduction rule

$$(\textsc{CaseApp}), \qquad \{\!|\theta|\!\}.(MN) \quad \to \quad (\{\!|\theta|\!\}.M)N.$$

where θ denotes a *case binding*, that is a finite map from constructors to terms (cf. Section 2.1). In practice, this rule pushes case analysis to the head of the destructured term before performing constructor substitution:

$$\begin{array}{lll} \{\!|c \mapsto M|\!\}.(cN_1 \cdots N_k) & \to^*_{\textsc{CaseApp}} & (\{\!|c \mapsto M|\!\}.c)N_1 \cdots N_k \\ & \to_{\textsc{CaseCons}} & MN_1 \cdots N_k \end{array}$$

(More examples of the uses of this rule will be given in Section 2.3.) Symmetrically, we introduce a reduction rule

$$(\textsc{CaseLam}) \qquad \{\!|\theta|\!\}.(\lambda x.N) \quad \to \quad \lambda x.(\{\!|\theta|\!\}.M) \qquad (x \notin FV(\theta))$$

[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* $(\{\!|\theta|\!\}.M)N = \{\!|\theta N|\!\}.M$ that comes from logic, a rule which is well typed... but incompatible with the reduction rules of our calculus!

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

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 (Böhm 1979; Barendregt 1984). For this reason, the formalism provides a special constant written ✠ and called the *daimon* (following the terminology and notation of Girard 2001) that requests the termination of the programme – something like an `exit` system call – and which will be used as the main technical device to observe normal forms and separate them.

This article is an extension of Arbiser *et al.* (2006).

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 $\mathscr{V}$ and $\mathscr{C}$, respectively. In what follows, we assume that both sets $\mathscr{V}$ and $\mathscr{C}$ 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)
		$\mid$	c	(Constructor)
		$\mid$	✠	(Daimon)
		$\mid$	MN	(Application)
		$\mid$	$\lambda x.M$	(Abstraction)
		$\mid$	$\{\!\|\theta\|\!\}.M$	(Case construct)
Case bindings	θ, ϕ	$::=$	$c_1 \mapsto M_1; \ldots; c_n \mapsto M_n$	($c_i \neq c_j$ for $i \neq j$)

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

Constructor binding Each case binding θ is formed as a 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 *domain* of a case binding $\theta = (c_1 \mapsto M_1; \ldots; c_n \mapsto M_n)$ is defined as $dom(\theta) = \{c_1, \ldots, c_n\}$.

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

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

$$\theta \circ \phi \equiv \theta \circ (c_1 \mapsto M_1; \ldots; c_n \mapsto M_n) \quad \equiv \quad c_1 \mapsto \{\!|\theta|\!\}.\, M_1; \ldots; c_n \mapsto \{\!|\theta|\!\}.\, M_n$$

Notice that this operation is not syntactically associative, since

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

whereas

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

However, composition of case bindings only makes sense in the presence of the case conversion reduction rule $\{\!|\theta|\!\}.\, \{\!|\phi|\!\}.\, M \to \{\!|\theta \circ \phi|\!\}.\, M$ (see Section 2.2) that makes the two terms 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:

$$FV((c_i \mapsto M_i)_{i=1..n}) = FV(M_1) \cup \cdots \cup FV(M_n)$$
$$FV(\{\!|\theta|\!\}.\, M) = FV(\theta) \cup FV(M).$$

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:

$$(c_i \mapsto M_i)_{i=1..n}\{x := N\} = (c_i \mapsto M_i\{x := N\})_{i=1..n}$$
$$(\{\!|\theta|\!\}.\, M)\{x := N\} = \{\!|\theta\{x := N\}|\!\}.\, M\{x := N\}.$$

2.2 Reduction rules

The λ-calculus with constructors has nine primitive reduction rules that are depicted in Figure 1. These rules contain the usual β (now called AppLam) and η (now called LamApp) reduction rules of the $\lambda(\eta)$-calculus. Case analysis is propagated through terms using the rules CaseApp and CaseLam, and constructor substitution is performed by the rule CaseCons. There is also a rule CaseCase that composes adjacent case constructs – thus performing an identification that is essential to achieve separation.[2] Finally, the daimon ✠ comes with three reduction rules LamDai, AppDai and CaseDai describing how the constant ✠ destructs its evaluation context. (This will be given a formal meaning in Section 5.2, Lemma 24.) Note that the

[2] Unlike the CaseApp reduction rule, the CaseCase reduction rule is really a commutative conversion rule.

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

Beta-reduction					
AppLam	(AL)	$(\lambda x.\, M)N$	$\to$	$M\{x := N\}$	
AppDai	(AD)	$\maltese N$	$\to$	$\maltese$	
Eta-reduction					
LamApp	(LA)	$\lambda x.\, Mx$	$\to$	M	$(x \notin FV(M))$
LamDai	(LD)	$\lambda x.\, \maltese$	$\to$	$\maltese$	
Case propagation					
CaseCons	(CO)	$\{\!\{\theta\}\!\}.\, c$	$\to$	M	$((c \mapsto M) \in \theta)$
CaseDai	(CD)	$\{\!\{\theta\}\!\}.\, \maltese$	$\to$	$\maltese$	
CaseApp	(CA)	$\{\!\{\theta\}\!\}.\, (MN)$	$\to$	$(\{\!\{\theta\}\!\}.\, M)N$	
CaseLam	(CL)	$\{\!\{\theta\}\!\}.\, \lambda x.\, M$	$\to$	$\lambda x.\, \{\!\{\theta\}\!\}.\, M$	$(x \notin FV(\theta))$
Case conversion					
CaseCase	(CC)	$\{\!\{\theta\}\!\}.\, (\{\!\{\phi\}\!\}.\, M)$	$\to$	$\{\!\{\theta \circ \phi\}\!\}.\, M$	

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

daimon can be understood as an exception, but only as an exception *that cannot be caught* – a feature that is essential in the proof of separation. In a domain-theoretic setting, the daimon would naturally be interpreted as the top element $\top$ (maximal information, immediate termination), the opposite of Scott's bottom $\bot$ (undefinedness, non-termination).

In what follows, we will be interested not only in the system induced by the nine reduction rules taken together, but more generally in the 512 subsystems formed by all subsets of the nine reduction rules. (Analysing the commutation and composition properties of all these subsystems is the key ingredient of the confluence proof we will present in Section 4.) We denote by $\lambda\mathscr{B}_{\mathscr{C}}$ the calculus generated by all rules of Figure 1, and we write $\mathscr{B}_{\mathscr{C}}$ the sub-calculus generated by all rules but AppLam (a.k.a. β).

Notice that AppLam (β) and LamApp (η) are the only reduction rules that may apply to an ordinary λ-term in $\lambda\mathscr{B}_{\mathscr{C}}$.

2.3 Examples

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

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

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

From the CaseCons reduction rule of the calculus, we have

$$\begin{array}{rcl} \texttt{if true then } M_1 \texttt{ else } M_2 & \equiv & \{\!|\mathsf{true} \mapsto M_1;\ \mathsf{false} \mapsto M_2|\!\}.\mathsf{true} \ \rightarrow\ M_1 \\ \texttt{if false then } M_1 \texttt{ else } M_2 & \equiv & \{\!|\mathsf{true} \mapsto M_1;\ \mathsf{false} \mapsto M_2|\!\}.\mathsf{false} \ \rightarrow\ M_2 \end{array}$$

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 $\mathsf{0}$ to $\mathsf{0}$ and $\mathsf{s}\,n$ to n. In $\lambda\mathscr{B}_{\mathscr{C}}$, this function is implemented as

$$\mathrm{pred} \equiv \lambda n.\{\!|\mathsf{0} \mapsto \mathsf{0};\ \mathsf{s} \mapsto \lambda z.z|\!\}.n$$

(where $\mathsf{0}$ and s are distinct constructors). The computation of $\mathrm{pred}\,\mathsf{0}$ is obvious from the rules AppLam ($=\beta$) and CaseCons:

$$\mathrm{pred}\ \mathsf{0} \ \rightarrow\ \{\!|\mathsf{0} \mapsto \mathsf{0};\ \mathsf{s} \mapsto \lambda z.z|\!\}.\mathsf{0} \ \rightarrow\ \mathsf{0}.$$

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

$$\begin{array}{l} \mathrm{pred}\ (\mathsf{s}\ N) \rightarrow \{\!|\mathsf{0} \mapsto \mathsf{0};\ \mathsf{s} \mapsto \lambda z.z|\!\}.(\mathsf{s}\ N) \\ \qquad\qquad \rightarrow (\{\!|\mathsf{0} \mapsto \mathsf{0};\ \mathsf{s} \mapsto \lambda z.z|\!\}.\mathsf{s})\ N \ \rightarrow\ (\lambda z.z)\ N \ \rightarrow\ N \end{array}$$

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 $\lambda\mathscr{B}_{\mathscr{C}}$ as follows:

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

Pattern matching and recursion Recursion is implemented in $\lambda\mathscr{B}_{\mathscr{C}}$ in the same way as in the pure λ-calculus, by using a fixpoint combinator such as Church's Y or Turing's $\Theta \equiv (\lambda yf.f(yyf))\ (\lambda yf.f(yyf))$. Combining recursion with case analysis, one can define recursive functions on algebraic datatypes, such as for example the function 'append' that concatenates two lists

$$\mathrm{append} \ \equiv\ \Theta\ \big(\lambda f l_1 l_2.\{\!|\mathsf{nil} \mapsto l_2;\ \mathsf{cons} \mapsto \lambda x l'.\mathsf{cons}\ x\ (f\ l'\ l_2)|\!\}.l_1\big)$$

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

$$\begin{array}{rcl} \mathrm{plus} & \equiv & \Theta\ \big(\lambda f n m.\{\!|\mathsf{0} \mapsto m;\ \mathsf{s} \mapsto \lambda p.\mathsf{s}\ (f\ p\ m)|\!\}.n\big) \\ \mathrm{minus} & \equiv & \Theta\ \big(\lambda f n p.\{\!|\mathsf{0} \mapsto n;\ \mathsf{s} \mapsto \lambda q.f\ (\mathrm{pred}\ n)\ q)|\!\}.p\big) \end{array}$$

Variadic constructors Since constructors have no fixed arity in $\lambda\mathscr{B}_{\mathscr{C}}$, 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:

$$(v_1,\ldots,v_n) \ \equiv\ \mathsf{vec}\ x_1\ \cdots\ x_n.$$

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

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

$$\text{vappend} \quad \equiv \quad \lambda v_1 v_2.\, \{\!|\mathsf{vec} \mapsto v_1|\!\}.\, v_2,$$

but not the function accessing an element of a vector given by its index. Indeed, accessing the ith element x_i of a vector $v = \mathsf{vec}\ x_1 \cdots x_n$ 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 $\lambda\mathscr{B}_{\mathscr{C}}$ 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

$$(v_1, \ldots, v_n) \quad \equiv \quad \mathsf{vec}\ \bar{n}\ x_1\ \cdots\ x_n.$$

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

$$\begin{aligned}
\text{skip} &\equiv \Theta\ \left(\lambda f n x.\, \{\!|\mathsf{0} \mapsto x;\ \mathsf{s} \mapsto \lambda p z.\, f\, p\, x|\!\}.\, n\right)\\
\text{length} &\equiv \lambda v.\, \{\!|\mathsf{vec} \mapsto \lambda n.\, \text{skip}\ n\ n|\!\}.\, v\\
\text{access} &\equiv \lambda v i.\, \{\!|\mathsf{vec} \mapsto \lambda n.\, \text{skip}\ i\ (\lambda x.\, \text{skip}\ (\text{minus}\ n\ (\mathsf{s}\ i))\ x)|\!\}.\, v
\end{aligned}$$

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

$$\text{vappend} \quad \equiv \quad \lambda v_1 v_2.\, \{\!|\mathsf{vec} \mapsto \lambda n_1.\, \{\!|\mathsf{vec} \mapsto \lambda n_2.\, \mathsf{vec}\ (\text{plus}\ n_1\ n_2)|\!\}.\, v_2|\!\}.\, v_1$$

3 Preliminary results

Before proving the Church–Rosser property (Section 4), we first state some basic definitions and results.

3.1 General definitions and commutation results

We first recall some classic definitions.

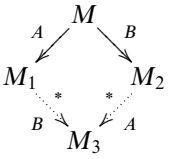
Definition 1
An Abstract Rewriting System (ARS) is a pair $A = (|A|, \to_A)$ formed by an arbitrary set $|A|$ (called the carrier of A) equipped with a binary relation $\to_A$ on $|A|$. We denote by $\to_A^=$ the reflexive closure of $\to_A$, by $\to_A^*$ the reflexive-transitive closure of $\to_A$,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

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

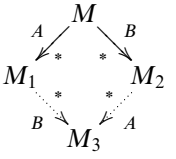
Definition 3
Let $(S, \to_A)$ and $(S, \to_B)$ be two ARSs defined on the same set. We say that:

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

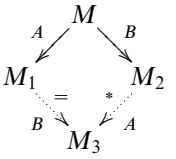
— *A weakly commutes* with B, written $A \mathbin{//_w} B$, if for all M, M_1, M_2 such that $M \to_A M_1$ and $M \to_B M_2$ there exists M_3 such that $M_1 \to_B^* M_3$ and $M_2 \to_A^* M_3$. In other words, the following diagram holds:



— *A commutes* with B, written $A \mathbin{//} B$, if for all M, M_1, M_2 such that $M \to_A^* M_1$ and $M \to_B^* M_2$ there exists M_3 such that $M_1 \to_B^* M_3$ and $M_2 \to_A^* M_3$. In other words, the following diagram holds:



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



Note that strong commutation is not a symmetrical relation, while commutation is.

Definition 4
Let A be an ARS. We say that

— A is *weakly confluent*, *locally confluent* or *weakly Church–Rosser (WCR)*, if $A \mathbin{//_w} A$.
— A is *confluent*, or *Church–Rosser (CR)*, if $A \mathbin{//} A$.
— A enjoys the *diamond property* if for all M, M_1, M_2 such that $M \to_A M_1$ and $M \to_A M_2$ there exists M_3 such that $M_1 \to_A M_3$ and $M_2 \to_A M_3$.

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 $\lambda\mathscr{B}_{\mathscr{C}}$ relies on standard results of rewriting (Baader & Nipkow 1999; Terese 2003), and in particular on the following lemmas:

Lemma 1
Let A, B, C be ARSs.

1. If $A \mathbin{//_w} B$ and $A \mathbin{//_w} C$ then $A \mathbin{//_w} B + C$.
2. If $A \mathbin{//} B$ and $A \mathbin{//} C$ then $A \mathbin{//} B + C$.

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

Proof
The first item is immediate from the definitions. The second item is proved by induction on the length of the $B + C$ reduction. □

Lemma 2 (*Newman's Lemma*)
Let A be an ARS. If A is weakly confluent and strongly normalising, then A is confluent.

Proof
See Baader & Nipkow (1999) and Terese (2003). □

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

Lemma 3
If $A /\!/_w B$ and $A + B$ is SN, then $A /\!/ B$.

Proof
Similar to Newman's Lemma, by well-founded induction. □

Lemma 4
If A strongly commutes with B, then $A /\!/ B$.

Proof
See Baader & Nipkow (1999) and Terese (2003). □

3.2 Free variables and substitution

The following lemmas are straightforward extensions of standard lemmas of the λ-calculus to the λ-calculus with constructors.

Lemma 5
Let M be a term or a case binding, P a term and x a variable. Then $FV(M\{x := P\}) \subseteq (FV(M) - \{x\}) \cup FV(P)$.

Proof
By a straightforward induction on M. □

Lemma 6
Let $P, Q \in \Lambda_{\mathscr{C}} + \mathscr{B}$. If $P \to_{\lambda\mathscr{B}_{\mathscr{C}}} Q$, then $FV(Q) \subseteq FV(P)$.

Proof
By induction on P, analysing each one of the rules. Note that rules AppLam, AppDai, CaseCons and CaseDai may remove free variables, while the rest of the rules will not. Also note that the case of rule CaseCase relies on the fact that $FV(\theta \circ \phi) \subseteq FV(\theta) \cup FV(\phi)$. □

Lemma 7
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 a straightforward induction on M. □

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

Lemma 8 (*Substitution lemma*)
For all terms and case bindings M, for all terms P, Q and variables x, 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 (cf. appendices). □

The following lemma expresses that substitution is preserved under each reduction rule of the calculus:

Lemma 9
Let R be one of the rules AppLam, AppDai, LamApp, LamDai, CaseLam, CaseApp, CaseDai, CaseCons, CaseCase.

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

Proof
By a straightforward induction on M. When $R =$ CaseCase, we use the fact that $(\theta \circ \phi)\{y := P\} = \theta\{y := P\} \circ \phi\{y := P\}$. □

This lemma immediately extends to many-steps reduction:

Corollary 1
Let R be one of the rules AppLam, AppDai, LamApp, LamDai, CaseLam, CaseApp, CaseDai, CaseCons, CaseCase.

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

3.3 Strong normalization of the $\mathscr{B}_{\mathscr{C}}$-calculus

We now prove that the sub-calculus $\mathscr{B}_{\mathscr{C}} = (\lambda\mathscr{B}_{\mathscr{C}} \setminus$ 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 the $\mathscr{B}_{\mathscr{C}}$-calculus*)
The $\mathscr{B}_{\mathscr{C}}$-calculus is SN.

Proof
See the appendices. □

Corollary 2 (*SN of $\lambda\mathscr{B}_{\mathscr{C}}$-subsystems*)
For every subsystem s of the $\lambda\mathscr{B}_{\mathscr{C}}$-calculus (induced by a subset of the nine primitive rules), s is SN if AppLam $\notin s$.

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

4 The Church–Rosser property

4.1 Critical pairs and closure conditions

Each of the nine primitive reduction rules of $\lambda\mathscr{B}_{\mathscr{C}}$ 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 Figures 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 Figures 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 six critical pairs (2), (4), (5), (6), (7) and (8) of Figure 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 nine rules given in Figure 1 fulfils the *closure conditions* and write $s \models \text{CC}$ if:

(CC1)	AppLam	$\in s$	$\wedge$	LamDai	$\in s$	$\Rightarrow$ AppDai $\in s$
(CC2)	LamApp	$\in s$	$\wedge$	AppDai	$\in s$	$\Rightarrow$ LamDai $\in s$
(CC3)	CaseApp	$\in s$	$\wedge$	AppLam	$\in s$	$\Rightarrow$ CaseLam $\in s$
(CC4)	CaseApp	$\in s$	$\wedge$	AppDai	$\in s$	$\Rightarrow$ CaseDai $\in s$
(CC5)	CaseLam	$\in s$	$\wedge$	LamApp	$\in s$	$\Rightarrow$ CaseApp $\in s$
(CC6)	CaseLam	$\in s$	$\wedge$	LamDai	$\in s$	$\Rightarrow$ CaseDai $\in s$

Intuitively, a subset that fulfils the six 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 a formal statement:

Theorem 1 (*Church–Rosser*)
For each of the 512 subsystems s of $\lambda\mathscr{B}_{\mathscr{C}}$ 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. $\lambda\mathscr{B}_{\mathscr{C}}$) obviously fulfils all closure conditions, we will get as an immediate corollary:

Corollary 3 (*Church–Rosser*)
$\lambda\mathscr{B}_{\mathscr{C}}$ is confluent.

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

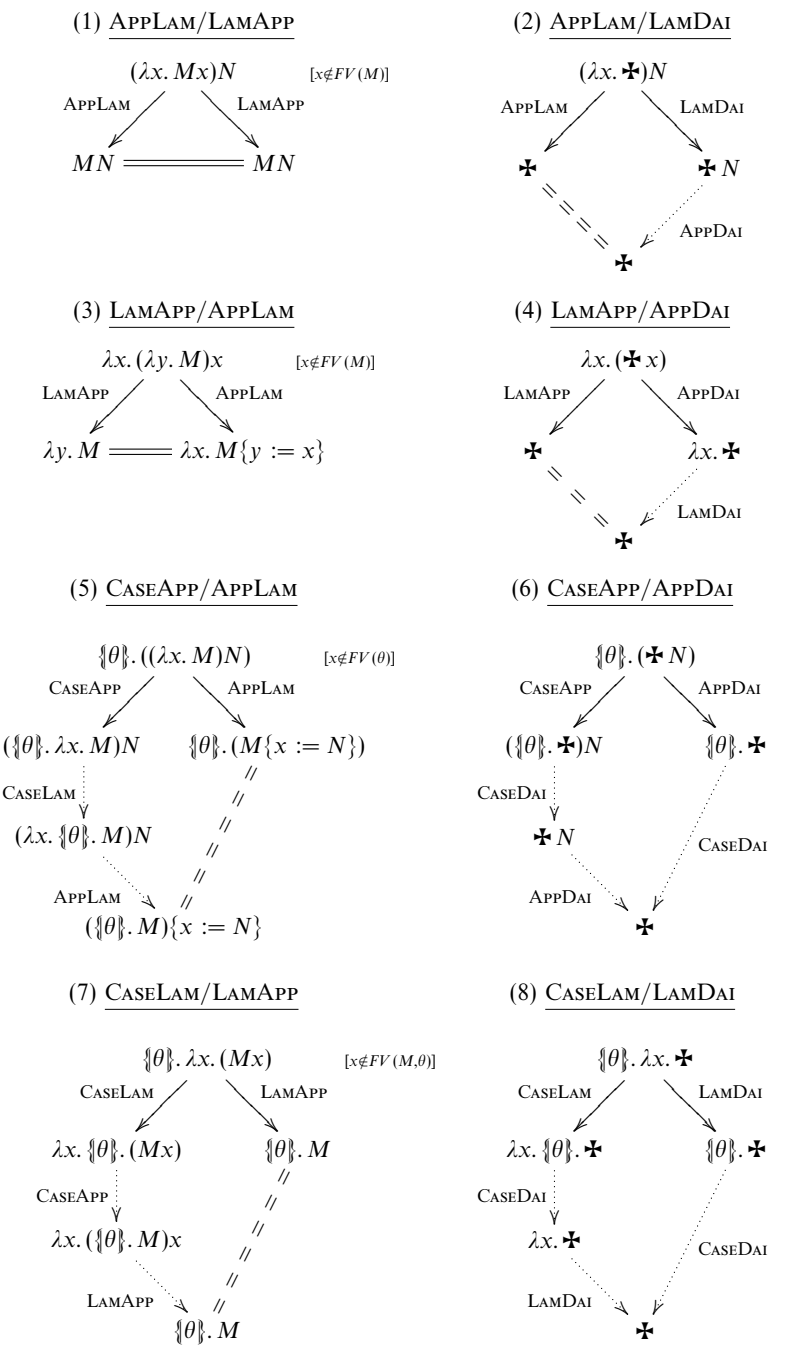


Fig. 2. Critical pairs 1–8 (/13).

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

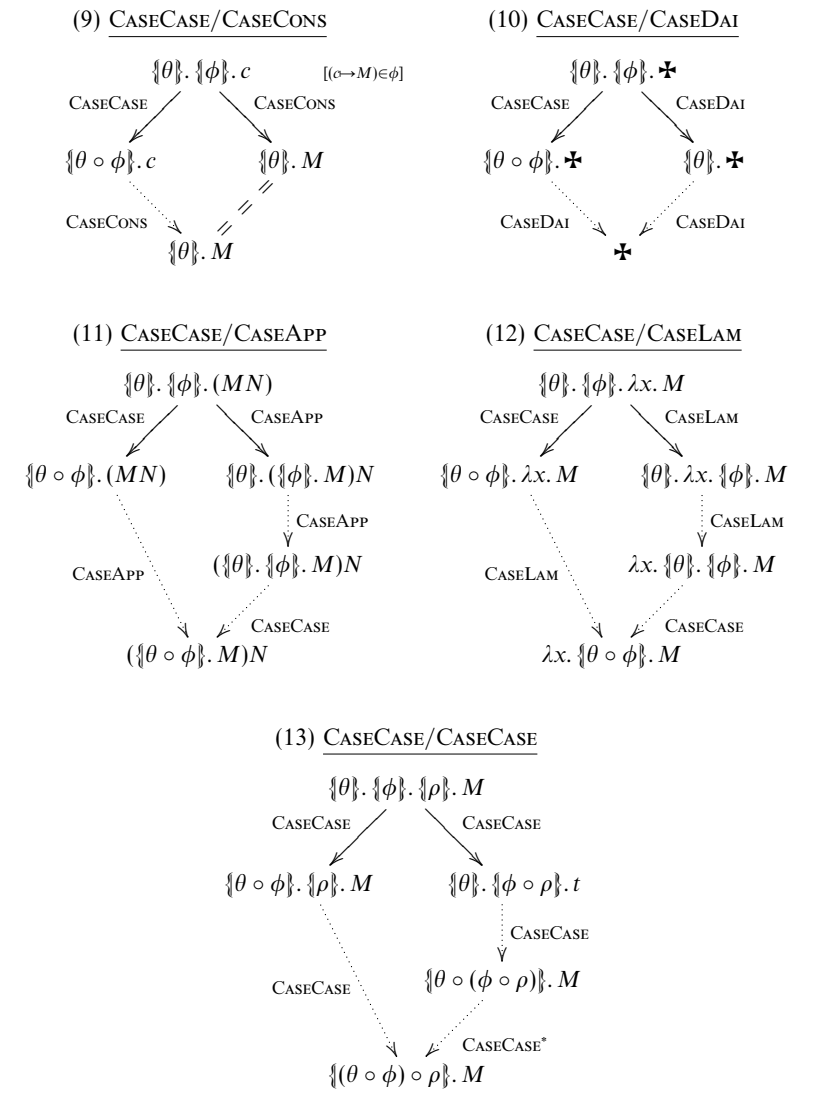


Fig. 3. Critical pairs 9–13 (/13).

The proof of Theorem 1 relies on a systematic analysis of the commutation properties of all pairs of subsystems (s_1, s_2) of $\lambda\mathscr{B}_{\mathscr{C}}$. 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 $(s_1, s_2) \models \mathrm{BCC}$ if

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

(BCC1) $\textsc{AppLam} \in s_1 \wedge \textsc{LamDai} \in s_2 \Rightarrow \textsc{AppDai} \in s_1$
(BCC2) $\textsc{LamApp} \in s_1 \wedge \textsc{AppDai} \in s_2 \Rightarrow \textsc{LamDai} \in s_1$
(BCC3) $\textsc{CaseApp} \in s_1 \wedge \textsc{AppLam} \in s_2 \Rightarrow \textsc{CaseLam} \in s_2$
(BCC4) $\textsc{CaseApp} \in s_1 \wedge \textsc{AppDai} \in s_2 \Rightarrow \textsc{CaseDai} \in (s_1 \cap s_2)$
(BCC5) $\textsc{CaseLam} \in s_1 \wedge \textsc{LamApp} \in s_2 \Rightarrow \textsc{CaseApp} \in s_2$
(BCC6) $\textsc{CaseLam} \in s_1 \wedge \textsc{LamDai} \in s_2 \Rightarrow \textsc{CaseDai} \in (s_1 \cap s_2)$
(BCC7) $\textsc{CaseCase} \in s_1 \wedge \textsc{CaseDai} \in s_2 \Rightarrow \textsc{CaseDai} \in s_1$
(BCC8) $\textsc{CaseCase} \in s_1 \wedge \textsc{CaseApp} \in s_2 \Rightarrow \textsc{CaseApp} \in s_1$
(BCC9) $\textsc{CaseCase} \in s_1 \wedge \textsc{CaseLam} \in s_2 \Rightarrow \textsc{CaseLam} \in s_1$

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

Again, the nine binary closure conditions come from an analysis of critical pairs. For example (BCC1) comes from the observation that critical pair (2) of Figure 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 every subsystem s of $\lambda\mathscr{B}_{\mathscr{C}}$: $s \models \text{CC}$ if $(s, s) \models \text{BCC}$.

We first show that:

Proposition 2

For every pair (s_1, s_2) of subsystems of $\lambda\mathscr{B}_{\mathscr{C}}$ 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 $\lambda\mathscr{B}_{\mathscr{C}}$ that fulfils the binary closure conditions. By induction on M we show that if $M \to_{r_1} M_1$ and $M \to_{r_2} M_2$ (for some $r_1 \in s_1$ and $r_2 \in s_2$), then there is a term M_3 such that $M_1 \to_{s_2}^* M_3$ and $M_2 \to_{s_1}^* M_3$. 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 $\lambda\mathscr{B}_{\mathscr{C}}$ that does not fulfil (at least) one of conditions (BCC1)–(BCC9). We easily build a counter-example from the originating critical pair. □

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

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 $\lambda\mathscr{B}_{\mathscr{C}}$[3]. From Proposition 2 and Corollary 2, we have:

$$\begin{array}{ccc} s_1 \,//_w\, s_2 & \Leftrightarrow & (s_1, s_2) \models \text{BCC} \\ (s_1 + s_2) \text{ is SN} & \Leftrightarrow & \textsc{AppLam} \notin (s_1 + s_2) \end{array}$$

for all pairs (s_1, s_2) of subsystems of $\lambda\mathscr{B}_{\mathscr{C}}$. 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 programme, 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 3, 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 + ¬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 1:

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

in order to propagate the knowledge of pairs of commuting subsystems (using the algorithm described in Figure 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 Figure 4. □

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

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

Table 1. *The 12 initial commutation lemmas*

(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

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**
  $C[s_1, s_2] := check\_bcc(s_1, s_2) \wedge \text{AppLam} \notin (s_1 \cup s_2)$
done;

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

[Close matrix using Lemma 1, item 2]
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 $C[s_1, (s_2 \cup s_3)] :=$ **true** **done**;

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

Fig. 4. Algorithm to build the commutation matrix.

The rest of this section is thus devoted to the proof of the 12 commutation properties of Table 1. Appendix B shows the complete deduction of confluence of the whole calculus from these 12 basic properties.

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

Remarks

The algorithm of Figure 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 $A \mathbin{//} B$ and $A \mathbin{//} C$, then $A \mathbin{//} (B + C)$'. (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 (Ríos 1993) – with the benefit that commutation and confluence results are obtained for all subsystems as well.

4.3 Proof of the 12 initial commutation properties

The first commutation property of Table 1 expresses the confluence of the reduction rule AppLam. As usual, we prove it (following Tait and Martin-Löf) by introducing the corresponding notion of parallel reduction:

Definition 7

The relations of *parallel* AppLam*-reduction* on terms and on case bindings (both written $\Rightarrow$) are defined as follows:

$$\frac{}{M \Rightarrow M}\ (\text{pRef}) \qquad \frac{M \Rightarrow M' \quad N \Rightarrow N'}{(\lambda x.\, M)N \Rightarrow M'\{x := N'\}}\ (\text{pAppLam})$$

$$\frac{M \Rightarrow M'}{\lambda x.\, M \Rightarrow \lambda x.\, M'}\ (\text{pLam}) \qquad \frac{M \Rightarrow M' \quad N \Rightarrow N'}{MN \Rightarrow M'N'}\ (\text{pApp})$$

$$\frac{M \Rightarrow M' \quad \theta \Rightarrow \theta'}{\{\!|\theta|\!\}.\, M \Rightarrow \{\!|\theta'|\!\}.\, M'}\ (\text{pCase})$$

$$\frac{M_1 \Rightarrow M_1' \quad \cdots \quad M_n \Rightarrow M_n'}{(c_i \mapsto M_i)_{i=1..n} \Rightarrow (c_i \mapsto M_i')_{i=1..n}}\ (\text{pCBind})$$

As usual, we check the following Proposition. Its item 4 will be used in the proof of Lemma 11.

Proposition 3 (*Properties of* $\Rightarrow$)

1. If $M \to_\beta M'$, then $M \Rightarrow M'$ (i.e. $\to_\beta \subset \Rightarrow$)
2. If $M \Rightarrow M'$, then $M \to_\beta^* M'$ (i.e. $\Rightarrow \subset \to_\beta^*$)
3. If $M \Rightarrow M'$ and $N \Rightarrow N'$, then $M\{x := N\} \Rightarrow M'\{x := N'\}$
4. If $\theta \Rightarrow \theta'$ and $\phi \Rightarrow \phi'$, then $\theta \circ \phi \Rightarrow \theta' \circ \phi'$
5. If $M \Rightarrow M_1$ and $M \Rightarrow M_2$, then
   there exists M_3 s.t. $M_1 \Rightarrow M_3$ and $M_2 \Rightarrow M_3$ (diamond property)

Proof

Item 1: by induction on the derivation of $M \to_\beta M'$ (cf. Appendix A). Item 2: by induction on the derivation of $M \Rightarrow M'$ (cf. Appendix A). Item 3: by induction

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

on the derivation of $M \Rightarrow M'$ (cf. Appendix A). Item 4: immediately follows from (pCase) (cf. Appendix A). Item 5: by induction on the derivation of $M \Rightarrow M_1$ (cf. Appendix A). □

From this we deduce that $\Rightarrow^* = \rightarrow_\beta^*$, and thus:

Proposition 4 (*1/12*)
AppLam // AppLam, i.e. AppLam is confluent.

The next five commutation properties (2–6) are of the form 'AppLam // r', where the reduction rule r is *linear*, that is, a rule that cannot duplicate subterms during contraction. (But it may erase subterms.)

Lemma 10
For each reduction rule

$$r \in \{\text{AppDai}; \text{LamApp}; \text{CaseCons}; \text{CaseDai}; \text{CaseLam}\},$$

the rule r strongly commutes with AppLam.

Proof
This is proved by a straightforward induction. Notice that the reduction rules AppDai, CaseCons, CaseDai and CaseLam induce no critical pair with AppLam. Only the rule LamApp induces critical pairs with AppLam, but these pairs are trivially closed (as in the pure λ-calculus). □

Proposition 5 (*2–6/12*)
For each reduction rule

$$r \in \{\text{AppDai}; \text{LamApp}; \text{CaseCons}; \text{CaseDai}; \text{CaseLam}\},$$

we have AppLam // r.

The commutation between AppLam and CaseCase is more delicate to handle since both rules may duplicate redexes of the other kind during contraction. However, the problem is greatly simplified if we replace AppLam by $\Rightarrow$ (parallel AppLam-reduction), since:

Lemma 11
CaseCase strongly commutes with $\Rightarrow$.

Proof
By induction on the derivation of $\Rightarrow$ (cf. Appendix A). □

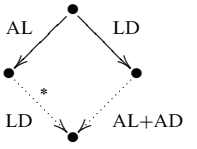
Proposition 6 (*7/12*)
AppLam // CaseCase.

Proof
By Lemma 4, we know that CaseCase commutes with $\Rightarrow$. But since we know that $\Rightarrow^* = \rightarrow_\beta^*$, we are done. □

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

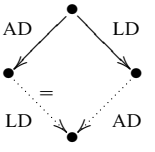
The following lemma describes the interaction between the reduction rules AppLam and LamDai, generalising critical pair (2) of Figure 2:

Lemma 12 (AppLam/LamDai)
The following diagram holds:



Proof
By structural induction on the initial term (top of the diagram). In all configurations where the initial AppLam- and LamDai-redexes are disjoint, contracting the AppLam-redex may duplicate the LamDai-redex (hence $\rightarrow_{LD}^{*}$ to close on the l.h.s.) whereas contracting the LamDai-redex leaves the AppLam-redex unaffected (hence $\rightarrow_{AL}$ to close on the r.h.s.). In the configuration of the critical pair AppLam/LamDai, we need no LamDai-reduction step to close on the l.h.s., but a single AppDai-reduction step to close on the r.h.s. (hence the use of AD). □

Lemma 13 (AppDai/LamDai)
The following diagram holds:



Proof
Obvious since both rules are linear and induce no critical pair. Notice that contracting the AppDai-redex may erase the LamDai-redex, hence the '=' to close on the l.h.s. □

The argument for the need of $\rightarrow^{=}$ in the previous proof will be used in the proof of other lemmas without explicit mention.

By merging the diagrams of Lemmas 12 and 13 we deduce:

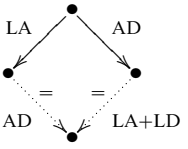
Lemma 14
LamDai strongly commutes with AppLam + AppDai.

Proposition 7 (*8/12*)
AppLam + AppDai // LamDai.

Again, the following lemma describes the interaction between the reduction rules LamApp and AppDai, generalising critical pair (4) of Figure 2:

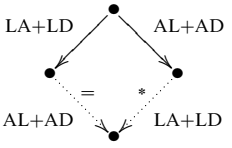
https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

Lemma 15 (LamApp/AppDai)
The following diagram holds:

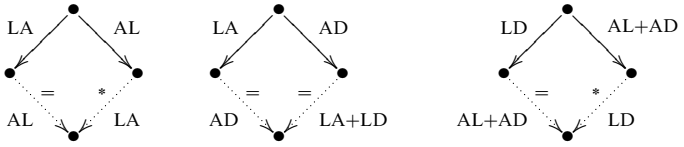


(The proof follows the same idea as for Lemma 12.)

Lemma 16
LamApp + LamDai strongly commutes with AppLam + AppDai:



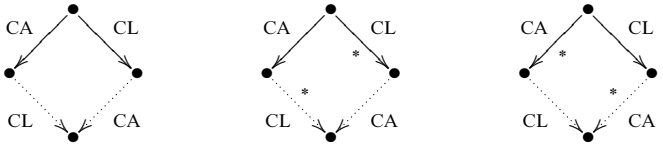
Proof
The proof proceeds by merging the following diagrams, that cover all the possible cases when $M \to_{\mathrm{LA+LD}} M_1$ and $M \to_{\mathrm{AL+AD}} M_2$:



The first diagram is Lemma 10 with rule r = LamApp, the second diagram is Lemma 15, and the third diagram is Lemma 14. □

Proposition 8 (*9/12*)
AppLam + AppDai // LamApp + LamDai.

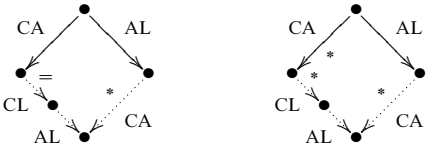
Lemma 17 (CaseApp/CaseLam)
The following diagrams hold:



Proof
The first diagram is proved by induction on the top term (there is no critical pair). The second (resp. third) diagram follows from diagram 1 by induction on the number of CaseLam (resp. CaseApp) reduction steps. □

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

Lemma 18 (CaseApp/AppLam)
The following diagrams hold:

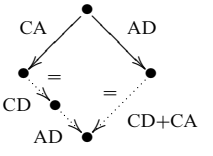


Proof
The first diagram is obtained by generalising critical pair (5) of Figure 2, following the spirit of Lemmas 12 and 15. The second diagram is deduced from the first, by induction on the number of CaseApp-reduction steps (top left), using the second diagram of Lemma 17 to close. □

Proposition 9 (*10/12*)
AppLam + CaseLam // CaseApp.

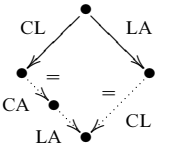
Proof
By induction on the number of (AppLam + CaseLam)-reduction steps, using the third diagram of Lemma 17 and the second diagram of Lemma 18. □

Lemma 19 (CaseApp/AppDai)
The following diagram holds:



Proof
This diagram is obtained by generalising critical pair (6) of Figure 2, following the spirit of Lemmas 12 and 15. □

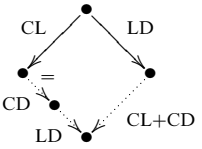
Lemma 20 (CaseLam/LamApp)
The following diagram holds:



Proof
This diagram is obtained by generalising critical pair (7) of Figure 2. □

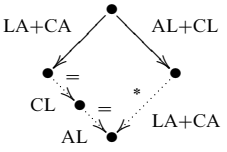
https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

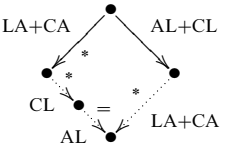
Lemma 21 (CaseLam/LamDai)
The following diagram holds:



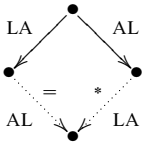
Proof
This diagram is obtained by generalising critical pair (8) of Figure 2. □

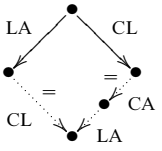
Lemma 22
The following diagrams hold:

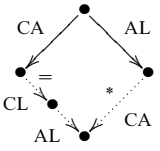


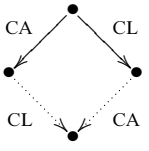


Proof
The first diagram is obtained by merging the following diagrams that cover all the possible cases:







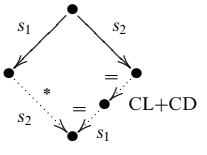


The diagrams above come from Lemmas 10, 20, 18 and 17, respectively. The second diagram is deduced from the first diagram, by induction on the number of (LamApp + CaseApp)-reduction steps (see Appendix A for the details). □

Proposition 10 (*11/12*)
AppLam + CaseLam // LamApp + CaseApp.

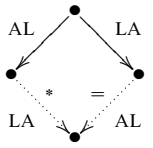
Proof
From the second diagram of Lemma 22, by induction on the number of (AppLam + CaseLam)-reduction steps. □

Lemma 23
Let s_1 = AppLam + AppDai + CaseDai + CaseLam and s_2 = LamApp + LamDai + CaseDai + CaseApp. The following diagram holds:

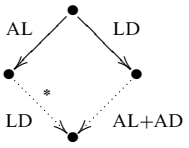


https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

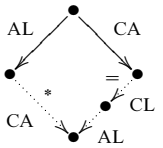
Proof
The proof proceeds by merging the following diagrams, that cover all the possible cases:



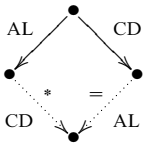
(Lemma 10)



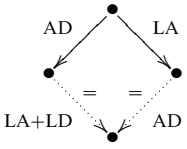
(Lemma 12)



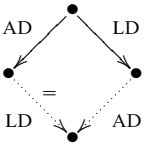
(Lemma 18)



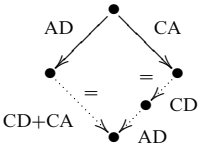
(Lemma 10)



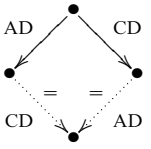
(Lemma 15)

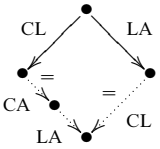


(Lemma 13)

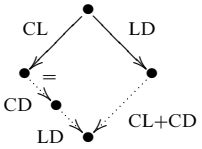


(Lemma 19)

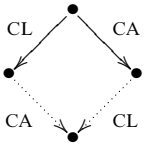




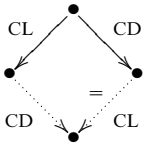
(Lemma 20)

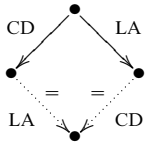


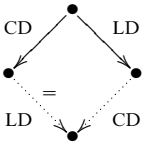
(Lemma 21)

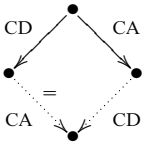


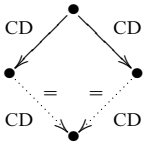
(Lemma 17)











(Non-annotated diagrams describe the interaction between two linear rules that have no critical pair.) □

Proposition 11 (*12/12*)
AppLam + AppDai + CaseDai + CaseLam commutes with LamApp + LamDai + CaseDai + CaseApp.

Proof
Again, let s_1 = AppLam + AppDai + CaseDai + CaseLam and s_2 = LamApp + LamDai + CaseDai + CaseApp. The proof of confluence is done by induction on the (s_2 + CL + CD)-depth of the top term, using Lemma 23 to close the diagram. □

5 Separation

In this section, we now prove the weak separation theorem for $\lambda\mathscr{B}_{\mathscr{C}}$, which basically expresses that two distinct normal forms of the calculus can be separated using a well chosen context. Since the calculus provides no mechanism to recover a pattern matching failure (represented by a subterm of the form $\{\!|\theta|\!\}.c$ where c is unbound

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

in θ), we have to exclude all such subterms from the separation process – which explains the distinction between *defined* and *undefined* normal forms we introduce in Section 5.1.

As for Böhm's theorem in the pure λ-calculus, the main technical difficulty is to separate head normal forms of different arities (i.e. starting with a different number of λ-abstractions). To solve this problem, we split the separation process in two steps. First, we define a notion of *disagreement* (at some depth d), and show that two distinct normal forms of the calculus can be η-expanded ('cooked') in such a way that they disagree at some depth (Lemma 26). The advantage of the notion of disagreement is that it maintains the consistency of arities of all head-normal forms that are encountered along the disagreement path, thus preparing both terms for separation. Once this property has been proved, it suffices to show that any pair of disagreeing terms can be separated using a suitable context (Prop. 13).

This section is organised as follows. In Section 5.1 we define the different notions of normal forms we will use throughout the proof, as well as the notions of defined and undefined terms. Separation contexts are introduced in Section 5.2, as well as the corresponding notion of observation. In Section 5.3 we introduce the standard encoding of tuples in the λ-calculus and study its interactions with the case construct, before showing how we technically separate distinct free variables in Section 5.4. The formal definition of disagreement is introduced in Section 5.5, with the corresponding cooking lemma (Lemma 26). Then we introduce the notion of parallel substitution in Section 5.6 before proving the main proposition (Prop. 13) in Section 5.7.

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:

Head term
$$H ::= x \mid c \mid \{\!\{\theta\}\!\}.\, x \mid \{\!\{\theta\}\!\}.\, c \quad (c \notin \mathrm{dom}(\theta))$$

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

Quasi-hnf
$$M ::= \maltese \mid \lambda x_1 \cdots x_n.\, H N_1 \cdots N_k$$

where H is an arbitrary head term, called the *head* of M, and where $N_1, \ldots, N_k$ are arbitrary terms ($k \geqslant 0$).

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

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

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

$$M \quad ::= \quad \maltese \quad | \quad \lambda x_1 \cdots x_n . H N_1 \cdots N_k \qquad \text{(where } H \text{ is defined)}$$

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

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

More generally, we call a *defined* term (resp. an *undefined* term) any term that reduces in zero or more steps 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 Section 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.

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 12
Quasi-normal terms, quasi-normal head terms, and quasi-normal case bindings are (mutually) characterised by the following BNF:

Q.n.-terms	N	$::=$	$\maltese \mid \lambda x_1 \cdots x_n . H N_1 \cdots N_k$	$(k \geqslant 0)$
Q.n.-head-terms	H	$::=$	$x \mid c \mid \{\!|\theta|\!\}.x \mid \{\!|\theta|\!\}.c$	$(c \notin \mathrm{dom}(\theta))$
Q.n.-case bind.	θ	$::=$	$c_1 \mapsto N_1; \ldots; c_p \mapsto N_p$	$(p \geqslant 0)$

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*:

Evaluation contexts $\qquad E[] \quad ::= \quad []N_1 \cdots N_n \quad | \quad \{\!|\theta|\!\}.[]N_1 \cdots N_n$

(The second form should be read $(\{\!|\theta|\!\}.[])N_1 \cdots N_n$.)

Notice that the composition $E'[E[]]$ of two evaluation contexts $E[]$ and $E'[]$ is not always an evaluation context, but that it always reduces to an evaluation context

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

using zero, one or several steps of the CaseApp rule, possibly followed by a single step of the CaseCase rule:

$$\begin{array}{rrcl}
 & [\quad []N_1 \cdots N_k \quad]N'_1 \cdots N'_{k'} & = & []N_1 \cdots N_k N'_1 \cdots N'_{k'} \\
 & [\quad \{\!|\theta|\!\}.\,[]N_1 \cdots N_k \quad]N'_1 \cdots N'_{k'} & = & \{\!|\theta|\!\}.\,[]N_1 \cdots N_k N'_1 \cdots N'_{k'} \\
\{\!|\theta'|\!\}.\,[& \quad []N_1 \cdots N_k \quad]N'_1 \cdots N'_{k'} & \to^* & \{\!|\theta'|\!\}.\,[]N_1 \cdots N_k N'_1 \cdots N'_{k'} \\
\{\!|\theta'|\!\}.\,[& \quad \{\!|\theta|\!\}.\,[]N_1 \cdots N_k \quad]N'_1 \cdots N'_{k'} & \to^* & \{\!|\theta' \circ \theta|\!\}.\,[]N_1 \cdots N_k N'_1 \cdots N'_{k'}
\end{array}$$

Remark 1
An evaluation context $E[]$ can always be regarded as a term (of a particular form) that contains exactly one occurrence of a distinguished variable depicted [] – the hole. In particular, since the unique occurrence of the hole [] in an evaluation context $E[]$ is outside the scope of all the binders of $E[]$, the operation of replacement $E[M]$ works just as the ordinary operation of substitution $E\{[] := M\}$ of λ-calculus. (This is of course not the case for the general notion of context with one hole – think of $C[x]$ where $C[] = \lambda x.\,[]$.)

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

Lemma 24
In any evaluation context $E[]$ one has $E[✠] \to^* ✠$.

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

Lemma 25
Given an undefined term U, the term $E[U]$ is undefined in any evaluation context $E[]$.

The daimon ✠ 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 $C[]$ such that either:
  - $C[M_1] \to^* ✠$ and $C[M_2]$ is undefined, or
  - $C[M_2] \to^* ✠$ and $C[M_1]$ is undefined;
- *strongly separable* if there exists two contexts $C_1[]$ and $C_2[]$ such that
  - $C_1[M_1] \to^* ✠$ and $C_1[M_2]$ is undefined, and
  - $C_2[M_2] \to^* ✠$ and $C_2[M_1]$ is undefined.

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

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

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 $c \notin \mathrm{dom}(\theta)$.

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 $\langle M_1;\ldots;M_n\rangle \equiv \lambda p.\, pM_1\cdots M_n$. 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:

$$\langle M_1;\ldots;M_k;*_{n-k}\rangle \;\equiv\; \lambda x_{k+1}\cdots x_n p.\, pM_1\cdots M_k x_{k+1}\cdots x_n \qquad (0\leqslant k\leqslant n)$$

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

$$\langle *_n\rangle M_1M_2\cdots M_n \;\to\; \langle M_1;*_{n-1}\rangle M_2\cdots M_n \;\to\; \cdots \;\to\; \langle M_1;M_2;\ldots;M_n\rangle,$$

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

$$\langle M_1;M_2;\ldots;M_n\rangle P \;\to\; PM_1M_2\cdots M_n.$$

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

$$\{\!|\theta|\!\}.\langle \vec{M};*\rangle \quad = \quad \{\!|\theta|\!\}.\lambda\vec{x}p.\, p\vec{M}\vec{x} \quad \to^* \quad \lambda\vec{x}p.\,\{\!|\theta|\!\}.\, p\vec{M}\vec{x},$$

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 a hybrid object formed as a combination of a case construct with a tuple – a *casuple* – that will be written:

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

Casuples work just as ordinary tuples

$$\begin{aligned}\{\!|\theta \mid *_n|\!\}M_1M_2\cdots M_n \;&\to\; \{\!|\theta \mid M_1;*_{n-1}|\!\}M_2\cdots M_n\\ &\to\; \cdots \;\to\; \{\!|\theta \mid M_1;M_2;\ldots;M_n|\!\},\end{aligned}$$

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:

$$\{\!|\theta \mid M_1;M_2;\ldots;M_n|\!\}P \quad \to \quad \{\!|\theta|\!\}.\, PM_1M_2\cdots M_n.$$

Casuples frequently appear in the process of separating terms (cf. Proposition 13 and its proof in Appendix A), typically when a variable x in a subterm of the form $(\{\!|\theta|\!\}.x)N_1\cdots N_k$ is substituted (during a β-contraction step) by a n-uple constructor sent by the separation context to analyse this subterm.

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

5.4 Encoding names

Separation of distinct free variables is achieved by substituting them with closed terms that can be easily separated. For that, we associate to every variable x a unique numeral $\mathsf{s}^n\mathsf{0}$ that we write $\mathtt{x}$ (using the same name written in typewriter face) and call the *symbol* of x.

The equality of two symbols $\mathtt{x}$ and $\mathtt{y}$ (associated to variables x and y) is easily tested using the equality test for natural numbers:

$$\begin{aligned}\mathsf{eq} \;\equiv\; \Theta(\lambda f xy.\, \{\!| \;\; & \mathsf{0} \mapsto \{\!|\mathsf{0} \mapsto \mathsf{true};\; \mathsf{s} \mapsto \lambda y'.\, \mathsf{false}|\!\}.\, y; \\ & s \mapsto \lambda x'.\, \{\!|\mathsf{0} \mapsto \mathsf{false};\; \mathsf{s} \mapsto \lambda y'.\, fx'y'|\!\}.\, y|\!\}.\, x)\end{aligned}$$

(where Θ is Turing's fixpoint combinator, cf. Section 2.3). We easily check that

$$\mathsf{eq}\,(\mathsf{s}^n\mathsf{0})\,(\mathsf{s}^m\mathsf{0}) \;\to^*\; \begin{cases}\mathsf{true} & \text{if } n = m \\ \mathsf{false} & \text{if } n \neq m\end{cases}$$

for all $n, m \in \mathbb{N}$, so that the term $\mathsf{eq}\,\mathtt{x}\,\mathtt{y}$ reduces to true if and only if x and y are the same variable, and it reduces to false otherwise.

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 $x_1, x_2 \in \mathscr{V}$, and $x_1 = x_2$; or
- $H_1 = c_1$ and $H_2 = c_2$ for some $c_1, c_2 \in \mathscr{C}$, 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 $x_1, x_2 \in \mathscr{V}$, and $\mathrm{dom}(\theta_1) = \mathrm{dom}(\theta_2)$ 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 $x_1, x_2 \in \mathscr{V}$, and either $x_1 \neq x_2$ or $\mathrm{dom}(\theta_1) \neq \mathrm{dom}(\theta_2)$.

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

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

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

– (*Base case*) We write $\mathrm{dis}_0(M_1, M_2)$ if either:
  - $M_1 = \maltese$ 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 $M_2 = \maltese$; 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 $d \in \mathbb{N}$, we write $\mathrm{dis}_{d+1}(M_1, M_2)$ 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 $c \in \mathrm{dom}(\theta_1) = \mathrm{dom}(\theta_2)$ such that $\mathrm{dis}_d(\theta_1(c), \theta_2(c))$; or
  - There is a position $1 \leqslant k \leqslant \min(k_1, k_2)$ such that $\mathrm{dis}_d(N_{1,k}, N_{2,k})$.

Lemma 26 (*Cooking lemma*)
If M_1 and M_2 are two completely defined normal terms (w.r.t. all the reduction rules including 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 $M_1' \to_\eta^* M_1$, $M_2' \to_\eta^* M_2$, and $\mathrm{dis}_d(M_1', M_2')$ for some $d \in \mathbb{N}$.

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. $M_1 = \maltese$ and M_2 is of the form $M_2 = \lambda x_1 \cdots x_n.\, H N_1 \cdots N_k$ (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
  $M_1 = \lambda x_1 \cdots x_{n_1}.\, H_1 N_{1,1} \cdots N_{1,k_1}$ and
  $M_2 = \lambda x_1 \cdots x_{n_2}.\, H_2 N_{2,1} \cdots N_{2,k_2}$.
  Without loss of generality, we can assume than $n_1 \leqslant n_2$, 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 $\mathrm{dom}(\theta_1) = \mathrm{dom}(\theta_2)$. 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 $P_i' \to_\eta^* P_i$ (for $i = 1, 2$) such that $\mathrm{dis}_d(P_1, P_2)$ for some $d \in \mathbb{N}$. 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 $M_i' \to_\eta^* M_i$ for $i = 1, 2$, and $\mathrm{dis}_{d+1}(M_1', M_2')$.
– Case 2.1.3. $H_1 = H_2 = H$. In this case, our initial assumption $M_1 \neq M_2$ expresses that the lists $(N_{1,1}, \ldots, N_{1,k_1})$ and $(N_{2,1}, \ldots, N_{2,k_2})$ differ. Again, we distinguish two cases:

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

– Case 2.1.3.1. There exists $1 \leqslant k \leqslant \min(k_1, k_2)$ such that $N_{1,k} \neq N_{2,k}$. By induction hypothesis, there are terms $N'_{i,k}$ $(i = 1, 2)$ such that $N'_{i,k} \to_\eta^* N_{i,k}$ and $\mathrm{dis}_d(N'_{1,k}, N'_{2,k})$ for some $d \in \mathbb{N}$. Let then define M'_i $(i = 1, 2)$ from M_i by replacing the subterm $N_{i,k}$ by $N'_{i,k}$. We have $M'_i \to_\eta^* M_i$ for $i = 1, 2$, and $\mathrm{dis}_{d+1}(M'_1, M'_2)$.
– Case 2.1.3.2. For all $1 \leqslant k \leqslant \min(k_1, k_2)$, 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 $N'_{1,k_1+1} \to_\eta^* y$, $N'_{2,k_1+1} \to_\eta^* N_{2,k_1+1}$ and $\mathrm{dis}_d(N'_{1,k_1+1}, N'_{2,k_1+1})$ for some $d \in \mathbb{N}$. Let us set $M'_1 = \lambda x_1 \cdots x_n y.\, H N_{1,1} \cdots N_{1,k_1} N'_{1,k_1+1}$ and
$M'_2 = \lambda x_1 \cdots x_n y.\, H N_{2,1} \cdots N_{2,k_1} N'_{2,k_1+1} N_{2,k_1+2} \cdots N_{2,k_2} y$.
By construction we have $M'_i \to_\eta M_i$ for $i = 1, 2$ and $\mathrm{dis}_{d+1}(M'_1, M'_2)$.
– 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
$M'_1 = \lambda x_1 \cdots x_{n_2}.\, H_1 N_{1,1} \cdots N_{1,k_1} x_{n_1+1} \cdots x_{n_2}$.
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 *(parallel) substitution*[4] is a finite set of pairs of the form

$$\sigma = \{(x_1 := N_1); \ldots; (x_n := N_n)\}$$

(where $x_1, \ldots, x_n$ are variables and $N_1, \ldots, N_n$ 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

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

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

$$\sigma_X^K \equiv \{(x_1 := \langle \mathtt{x}_1; *_K \rangle); \ldots; (x_n := \langle \mathtt{x}_n; *_K \rangle)\}$$

[4] The notion of parallel substitution should not be confused with the *sequential* substitution $M\{x_1 := N_1\} \cdots \{x_n := N_n\}$. However, both notions coincide in the case where $(FV(N_1) \cup \cdots \cup FV(N_n)) \cap \{x_1; \ldots; x_n\} = \emptyset$.

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

the substitution which maps each variable $x_i \in X$ to a $(K+1)$-uple constructor which is partially applied to the symbol $\mathtt{x}_i$ 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 $k \geqslant 0$ such that M has a subterm of the form $HN_1 \cdots N_k$.

Lemma 27
Let M be a defined quasi-head normal term which is not the daimon, that is, a term of the form

$$M \equiv \lambda x_1 \cdots x_n . H N_1 \cdots N_k$$

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

$$E_1\big[M[\sigma_X^K]\big] \to^* \maltese$$

and another closed evaluation context $E_2[]$ such that

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

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

1. $H = y$ (*variable*). Take

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

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

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

Similarly for E_2 we take $E_2[] \equiv []\langle \mathtt{x}_1; *_K\rangle \cdots \langle \mathtt{x}_n; *_K\rangle ?_{k+1} \cdots ?_K U$ where $?_{k+1}, \ldots, ?_K$ are arbitrary terms, and U any closed undefined head term.

2. $H = \{\!|\theta|\!\}.\, y$ (*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 $\maltese$ and by the undefined head term U.

3. $H = c$ (*constructor*). Set $E_1[] \equiv \{\!|c \mapsto \maltese|\!\}.\, []$. We check that

$$\begin{aligned} E_1\big[M[\sigma_X^K]\big] &= \{\!|c \mapsto \maltese|\!\}.\, (\lambda x_1 \cdots x_n . c N_1 \cdots N_k)[\sigma_X^K] \\ &= \big(\{\!|c \mapsto \maltese|\!\}.\, (\lambda x_1 \cdots x_n . c N_1 \cdots N_k)\big)[\sigma_X^K] \\ &\to^* \big(\lambda x_1 \cdots x_n . (\{\!|c \mapsto \maltese|\!\}.\, c) N_1 \cdots N_k\big)[\sigma_X^K] \\ &\to^* \big(\lambda x_1 \cdots x_n . \maltese N_1 \cdots N_k\big)[\sigma_X^K] \\ &\to^* \maltese \end{aligned}$$

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

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

Proposition 13 (*Separation of disagreeing terms*)
Let K be a natural number, and M_1, M_2 be 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 $d \in \mathbb{N}$. Then there exists a closed evaluation context $E[]$ such that either

- $E\left[M_1[\sigma_X^K]\right] \to^* \maltese$ and $E\left[M_2[\sigma_X^K]\right]$ is undefined, or
- $E\left[M_2[\sigma_X^K]\right] \to^* \maltese$ and $E\left[M_1[\sigma_X^K]\right]$ is undefined;

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 Section 5.6.

Proof
The proof proceeds by induction on the depth d of the disagreement between M_1 and M_2 (cf. appendices for the details). □

Theorem 2 (*Separation*)
Let M_1 and M_2 be completely defined terms in normal form. If $M_1 \neq M_2$, then M_1 and M_2 are weakly separable.

Proof
Assume that M_1 and M_2 are distinct normal terms. By the cooking lemma, there exist two completely defined quasi-normal terms M_1' and M_2' such that $M_i' \to_\eta^* M_i$ (for $i = 1, 2$) and $\mathrm{dis}_d(M_1', M_2')$ for some $d \in \mathbb{N}$. Now 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 a closed evaluation context $E[]$ such that

$$E\left[M_1'[\sigma_X^K]\right] \to^* \maltese \qquad \text{and} \qquad E\left[M_2'[\sigma_X^K]\right] \text{ undefined} \qquad \text{(or symmetrically)}$$

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 $x_1, \ldots, x_n$ are the elements of X) and we are done by Corollary 3 (Church–Rosser). □

6 Conclusion

We have introduced an extension of λ-calculus, $\lambda\mathscr{B}_{\mathscr{C}}$, 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 $\lambda\mathscr{B}_{\mathscr{C}}$ is confluent and 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 $\lambda\mathscr{B}_{\mathscr{C}}$. Surprisingly, the mechanical propagation rule

$$\text{'if } A \mathbin{/\!/} B \text{ and } A \mathbin{/\!/} C \text{ then } A \mathbin{/\!/} (B + C)\text{'}$$

(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

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

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 $\lambda\mathscr{B}_{\mathscr{C}}$ The separation theorem we proved suggests that head normal forms of $\lambda\mathscr{B}_{\mathscr{C}}$ could be the adequate bricks to define a notion of Böhm-tree (Böhm 1979; Barendregt 1984) for $\lambda\mathscr{B}_{\mathscr{C}}$ – 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 $\lambda\mathscr{B}_{\mathscr{C}}$.

Which type system for $\lambda\mathscr{B}_{\mathscr{C}}$? 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 $\lambda\mathscr{B}_{\mathscr{C}}$.

References

Arbiser, A., Miquel, A. & Ríos, A. (2006) A λ-calculus with constructors. In *Lecture Notes in Computer Science Volume 4098*. International Conference of Rewriting Techniques and Applications. Springer-Verlag.

Baader, F. & Nipkow, T. (1999) *Rewriting and All That*. Addison-Wesley.

Barendregt, H. (1984) The lambda calculus: Its syntax and semantics., In *Studies in Logic and The Foundations of Mathematics*, J. Barwise, D. Kaplan, H. J. Keisler, P. Suppes and A. S. Troelstra (eds), vol. 103. North-Holland.

Böhm, C., Dezani-Ciancaglini, M., Peretti, P. & Ronchi Della Rocha, S. (1979) A discrimination algorithm inside lambda-beta-calculus. *Theor. Comput. Sci.*, **8** (3): 265–291.

Cerrito, S. & Kesner, D. (1999) Pattern matching as cut elimination. In *Logics In Computer Science (LICS'99)*, IEEE Computer Society, pp. 98–108.

Church, A. (1941) The calculi of lambda-conversion. In *Annals of Mathematical Studies*, vol. 6. Princeton.

Cirstea, H. & Kirchner, C. (1998) Rho-calculus, the rewriting calculus. In *Fifth International Workshop on Constraints in Computational Logics*, Jerusalem, Israel.

Girard, J.-Y. (2001) Locus solum: From the rules of logic to the logic of rules. *Math. Struct. Comput. Sci.*, **11** (3): 301–506.

Hudak, P., Peyton-Jones, S. & Wadler, P. (1992) Report on the programming language Haskell, a non-strict, purely functional language (Version 1.2). Sigplan Notices.

Jay, C. B. (2004) The pattern calculus. *ACM Trans. Program. Lang. Syst.*, **26** (6): 911–937.

Jay, C. B. & Kesner, D. (2006) Pure pattern calculus. In *Lecture Notes in Computer Science*, Sestoft, P. (ed), vol. 3924. Springer, pp. 100–114.

Kahl, W. (2003) Basic pattern matching calculi: Syntax, reduction, confluence, and normalisation. In *Technical Report* 16, Software Quality Research Laboratory, McMaster University.

Leroy, X. *et al.* (2008) The Objective Caml system release 3.11. Documentation and user's manual. `http://caml.inria.fr/`.

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

Milner, R., Tofte, M. & Harper, R. (1990) *The Definition of Standard ML*. MIT Press.

Ríos, A. (1993) *Contribution à l'étude des λ-calculs avec substitutions explicites*. PhD thesis, Université Paris 7.

Terese. (2003) *Term Rewriting Systems*, Marc Bezem, Jan Willem Klop and Roel de Vrijer (eds), Cambridge University Press. Cambridge Tracts in Theoretical Computer Science **55**.

van Oostrom, V. (1990) Lambda calculus with patterns. In *Technical Report IR-228*, Vrije Universiteit.

Appendix A. Proofs

Proof of Proposition 1 p. 590 (Strong normalisation of the $\mathscr{B}_{\mathscr{C}}$-calculus).
We define a measure function h on both terms (notation: M, N, etc.) and case bindings (notation: θ, ϕ, etc.) of the $\mathscr{B}_{\mathscr{C}}$-calculus by letting

$$\begin{aligned} h(x) &= 1 & h(MN) &= h(M) + h(N) \\ h(c) &= 1 & h(\lambda x.\, M) &= h(M) + 1 \\ h(\maltese) &= 1 & h(\{\!|\theta|\!\}.\, M) &= (h(\theta) + 2)h(M) \end{aligned}$$

$$h(\theta) = \sum_{c \in \mathrm{dom}(\theta)} h(\theta(c))$$

Note that for all terms and case bindings M we have $h(M) \geqslant 1$.
First, we prove that for all case bindings θ, ϕ: $h(\theta \circ \phi) = (h(\theta) + 2) \times h(\phi)$. Let us write $\phi = \{c_i \mapsto N_i\}_{i=1}^{k}$ $(k \geqslant 0)$ and $\theta \circ \phi = \{c_i \mapsto \{\!|\theta|\!\}.\, N_i\}_{i=1}^{k}$. We have

$$\begin{aligned} h(\theta \circ \phi) &= \sum_{i=1}^{k} h(\{\!|\theta|\!\}.\, N_i) = \sum_{i=1}^{k} (h(\theta) + 2) \times h(N_i) \\ &= (h(\theta) + 2) \times \sum_{i=1}^{k} h(N_i) = (h(\theta) + 2) \times h(\phi). \end{aligned}$$

□

Now we prove that for every redex $M \triangleright M'$ (reduction at root) of the $\mathscr{B}_{\mathscr{C}}$-calculus, we have $h(M) > h(M')$. We distinguish the following eight cases:

– AppDai: $\maltese\, M \to \maltese$

$$h(\maltese\, M) = 1 + h(M) \geqslant 2 > 1 = h(\maltese).$$

– LamApp: $\lambda x.\, M\, x \to M \quad (x \notin FV(M))$

$$h(\lambda x.\, M\, x) = h(M) + 2 > h(M).$$

– LamDai: $\lambda x.\, \maltese \to \maltese$

$$h(\lambda x.\, \maltese) = 2 > 1 = h(\maltese).$$

– CaseCons: $\{\!|\theta|\!\}.\, c \to \theta(c) \quad (c \in \mathrm{dom}(\theta))$

$$h(\{\!|\theta|\!\}.\, c) = (h(\theta) + 2) \times 1 \geqslant h(\theta(c)) + 2 > h(\theta(c)).$$

– CaseDai: $\{\!|\theta|\!\}.\, \maltese \to \maltese$

$$h(\{\!|\theta|\!\}.\, \maltese) = (h(\theta) + 2) \times 1 \geqslant 2 > 1 = h(\maltese).$$

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

– CaseApp: $\{\!\{\theta\}\!\}.(MN) \to (\{\!\{\theta\}\!\}.M)N$

$$\begin{aligned} h(\{\!\{\theta\}\!\}.(MN)) &= (h(\theta)+2)\times(h(M)+h(N)) \\ &\geqslant (h(\theta)+2)\times h(M)+2h(N) \\ &> (h(\theta)+2)\times h(M)+h(N) \;=\; h((\{\!\{\theta\}\!\}.M)N) \end{aligned}$$

– CaseLam: $\{\!\{\theta\}\!\}.\lambda x.M \to \lambda x.\{\!\{\theta\}\!\}.M \quad (x \notin FV(\theta))$

$$\begin{aligned} h(\{\!\{\theta\}\!\}.\lambda x.M) &= (h(\theta)+2)\times(h(M)+1) \\ &\geqslant (h(\theta)+2)\times h(M)+2 \\ &> (h(\theta)+2)\times h(M)+1 \;=\; h(\lambda x.\{\!\{\theta\}\!\}.M) \end{aligned}$$

– CaseCase: $\{\!\{\theta\}\!\}.\{\!\{\phi\}\!\}.M \to \{\!\{\theta\circ\phi\}\!\}.M$

$$\begin{aligned} h(\{\!\{\theta\}\!\}.\{\!\{\phi\}\!\}.M) &= (h(\theta)+2)\times(h(\phi)+2)\times h(M) \\ &= \big((h(\theta)+2)\times h(\phi)+2h(\theta)+4\big)\times h(M) \\ &\geqslant ((h(\theta)+2)\times h(\phi)+4)\times h(M) \\ &> ((h(\theta)+2)\times h(\phi)+2)\times h(M) \\ &= (h(\theta\circ\phi)+2)\times h(M) \;=\; h(\{\!\{\theta\circ\phi\}\!\}.M). \end{aligned}$$ □

Finally, we prove that if $M \to M'$ (resp. $\theta \to \theta'$) in the $\mathscr{B}_{\mathscr{C}}$-calculus, then $h(M) > h(M')$ (resp. $h(\theta) > h(\theta')$). It is done by a straightforward induction on the following rules:

$$\frac{M\to M'}{M\triangleright M'} \qquad \frac{\lambda x.M\to\lambda x.M'}{M\to M'} \qquad \frac{MN\to M'N}{M\to M'} \qquad \frac{MN\to MN'}{N\to N'}$$

$$\frac{\{\!\{\theta\}\!\}.M\to\{\!\{\theta'\}\!\}.M}{\theta\to\theta'} \qquad \frac{\{\!\{\theta\}\!\}.M\to\{\!\{\theta\}\!\}.M'}{M\to M'} \qquad \frac{\{c_i\mapsto N_i\}_{i=1}^{k}\to\{c_i\mapsto N_i'\}_{i=1}^{k}}{N_{i_0}\to N_{i_0}' \quad N_i\equiv N_i' \quad (i\neq i_0)}$$ □

Proof of Lemma 8 p. 590 (Substitution Lemma).
By induction on M.

– If $M = x$ we have by Lemma 7 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 $M = \maltese$, we have that $\maltese = \maltese$.
– 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 $M = \{\!\{\theta\}\!\}.M_1$, $M\{x := P\}\{y := Q\} =$
$\{\!\{\theta\{x := P\}\{y := Q\}\}\!\}.M_1\{x := P\}\{y := Q\} =_{IH}$

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

$\{\!|\theta\{y := Q\}\{x := P\{y := Q\}\}|\!\}.\, M_1\{y := Q\}\{x := P\{y := Q\}\} = M\{y := Q\}\{x := P\{y := Q\}\}$.

□

Proof of Proposition 3 p. 597, item 1 ($\to_{\mathrm{AL}} \subseteq \Rightarrow$).
We show that $M \to_{\mathrm{AL}} N$ implies $M \Rightarrow N$ by induction on M.

- If $M = x, c, \maltese$, the result holds vacuously.
- If the reduction takes place at the root, say $(\lambda x.P)Q \to_{\mathrm{AL}} P\{x := Q\}$, then $(\lambda x.P)Q \Rightarrow P\{x := Q\}$ using $P \Rightarrow P$, $Q \Rightarrow Q$, Corollary 1 and (pRef).
- If $M = PQ \to_{\mathrm{AL}} P'Q = N$, by IH $P \Rightarrow P'$ so $M \Rightarrow N$ by (pApp).
- If $M = PQ \to_{\mathrm{AL}} PQ' = N$, analogous.
- If $M = \lambda x.P \to_{\mathrm{AL}} \lambda x.P' = N$, by IH $P \Rightarrow P'$ so $M \Rightarrow N$ by (pLam).
- If $M = \theta \to_{\mathrm{AL}} \theta'$, let $\theta = (c_i \mapsto M_i)_{i=1,\ldots,n}$ with $M_j \to_{\mathrm{AL}} M'_j$ for some $1 \leqslant j \leqslant n$ and $\theta' = (c_i \mapsto M'_i)_{i=1,\ldots,n}$ where $M'_i = M_i$ for $i \neq j$, then by IH $M_j \Rightarrow M'_j$ thus $\theta \Rightarrow \theta'$ by (pCBind).
- If $M = \{\!|\theta|\!\}.\, M_1 \to_{\mathrm{AL}} \{\!|\theta'|\!\}.\, M_1 = N$, by IH $\theta \Rightarrow \theta'$ so $M \Rightarrow N$ by (pCase).
- If $M = \{\!|\theta|\!\}.\, M_1 \to_{\mathrm{AL}} \{\!|\theta|\!\}.\, M'_1 = N$, by IH $M_1 \Rightarrow M'_1$ so $M \Rightarrow N$ by (pCase).

□

Proof of Proposition 3 p. 597, item 2 ($\Rightarrow \subseteq \overset{*}{\to}_{\mathrm{AL}}$).
We prove that $P \Rightarrow Q$ implies $P \overset{*}{\to}_{\mathrm{AL}} Q$ by induction on the derivation of $P \Rightarrow Q$:

- if (pRef) was applied, trivial
- for (pAppLam), $P = (\lambda x.M)N \to_{\mathrm{AL}} M\{x := N\} \overset{*}{\to}_{\mathrm{AL}} M'\{x := N'\} = Q$, since by IH $M \overset{*}{\to}_{\mathrm{AL}} M'$ and $N \overset{*}{\to}_{\mathrm{AL}} N'$, and using Corollary 1
- for (pApp), $P = MN \overset{*}{\to}_{\mathrm{AL}} M'N' = Q$ since by IH $M \overset{*}{\to}_{\mathrm{AL}} M'$ and $N \overset{*}{\to}_{\mathrm{AL}} N'$
- for (pLam), $P = \lambda x.M \overset{*}{\to}_{\mathrm{AL}} \lambda x.M' = Q$ since by IH $M \overset{*}{\to}_{\mathrm{AL}} M'$
- for (pCBind), $P = (c_i \mapsto M_i)_{i=1,\ldots,n} \overset{*}{\to}_{\mathrm{AL}} (c_i \mapsto M'_i)_{i=1,\ldots,n} = Q$ since by IH $M_i \overset{*}{\to}_{\mathrm{AL}} M'_i$ for $1 \leqslant i \leqslant n$
- for (pCase), $P = \{\!|\theta|\!\}.\, M \overset{*}{\to}_{\mathrm{AL}} \{\!|\theta'|\!\}.\, M' = Q$ since by IH $\theta \overset{*}{\to}_{\mathrm{AL}} \theta'$ and $M \overset{*}{\to}_{\mathrm{AL}} M'$.

□

Proof of Proposition 3 p. 597, item 3.
We first prove that, for every term and case binding M, for all terms P, Q and every variable y, if $P \Rightarrow Q$, then $M\{y := P\} \Rightarrow M\{y := Q\}$, by induction on M.

- If $M = y$, then we have $P \Rightarrow Q$.
- If $M = x \neq y$, then by (pRef) $x \Rightarrow x$.
- If $M = \maltese$, then by (pRef) $\maltese \Rightarrow \maltese$.
- If $M = c$ a constructor, then by (pRef) $c \Rightarrow c$.
- 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) $M\{y := P\} = \lambda x.M_1\{y := P\} \Rightarrow \lambda x.M_1\{y := Q\} = M\{y := Q\}$.

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

– If $M = \theta = (c_i \mapsto M_i)_{i=1,\ldots,n}$, then by IH and (pCBind)
$M\{y := P\} = (c_i \mapsto M_i\{y := P\})_{i=1,\ldots,n}$
$\Rightarrow (c_i \mapsto M_i\{y := Q\})_{i=1,\ldots,n} = \theta\{y := Q\}$.
– If $M = \{\!|\theta|\!\}.N$, then by IH and pCase
$M\{y := P\} = \{\!|\theta\{y := P\}|\!\}.N\{y := P\}$
$\Rightarrow \{\!|\theta\{y := Q\}|\!\}.N\{y := Q\} = M\{y := Q\}$.

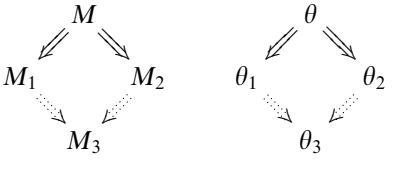
Now we generalize: for all terms and case bindings M, M', for all terms N, N' and every variable y, if $M \Rightarrow M'$ and $N \Rightarrow N'$, then $M\{y := N\} \Rightarrow M'\{y := N'\}$, by induction on the derivation of $M \Rightarrow M'$.

– if (pRef) was applied, $M = M'$, then by the above result $M\{y := N\} \Rightarrow M'\{y := N'\}$.
– for (pAppLam), $M = (\lambda x.P)Q$, $M' = P'\{x := Q'\}$ with $P \Rightarrow P'$ and $Q \Rightarrow Q'$, then $((\lambda x.P)Q)\{y := N\} = (\lambda x.P\{y := N\})Q\{y := N\}$. By IH, $P\{y := N\} \Rightarrow P'\{y := N'\}$ and $Q\{y := N\} \Rightarrow Q'\{y := N'\}$, thus
$(\lambda x.P\{y := N\})Q\{y := N\} \Rightarrow P'\{y := N'\}\{x := Q'\{y := N'\}\}$
$= P'\{x := Q'\}\{y := N'\}$ (by Lemma 8)
$= M'\{y := N'\}$ since x is fresh by the free variable convention.
– for (pApp), $M = PQ$, $M' = P'Q'$ with $P \Rightarrow P'$ and $Q \Rightarrow Q'$, then $(PQ)\{y := N\} = P\{y := N\}Q\{y := N\} \Rightarrow^{IH} P'\{y := N'\}Q'\{y := N'\} = (P'Q')\{y := N'\}$
– for (pLam), $(\lambda x.P)\{y := N\} = \lambda x.P\{y := N\} \Rightarrow^{IH} \lambda x.P'\{y := N'\} = (\lambda x.P')\{y := N'\}$
– for (pCase), $(c_i \mapsto M_i)_{i=1,\ldots,n}\{y := N\} = (c_i \mapsto M_i\{y := N\})_{i=1,\ldots,n} \Rightarrow^{IH} (c_i \mapsto M_i'\{y := N'\})_{i=1,\ldots,n} = (c_i \mapsto M_i')_{i=1,\ldots,n}\{y := N'\}$
– for (pCBind), $(\{\!|\theta|\!\}.P)\{y := N\} = \{\!|\theta\{y := N\}|\!\}.P\{y := N\} \Rightarrow^{IH}$
$\{\!|\theta'\{y := N'\}|\!\}.P'\{y := N'\} = (\{\!|\theta'|\!\}.P')\{y := N'\}$

□

Proof of Proposition 3 p. 597, item 4.
We prove that for all case bindings $\theta, \theta', \phi, \phi'$, if $\theta \Rightarrow \theta'$ and $\phi \Rightarrow \phi'$ then $\theta \circ \phi \Rightarrow \theta' \circ \phi'$. Let $\phi = (d_i \mapsto N_i)_{i=1,\ldots,n} \Rightarrow (d_i \mapsto N_i')_{i=1,\ldots,n} = \phi'$ with $N_i \Rightarrow N_i'$ for all $1 \leqslant i \leqslant n$. Then $\theta \circ \phi = (d_i \mapsto \{\!|\theta|\!\}.N_i)_{i=1,\ldots,n} \Rightarrow (d_i \mapsto \{\!|\theta'|\!\}.N_i')_{i=1,\ldots,n} = \theta' \circ \phi'$. □

Proof of Proposition 3 p. 597, item 5 (diamond property).
We prove that the following diagrams hold (for terms and case bindings, respectively)

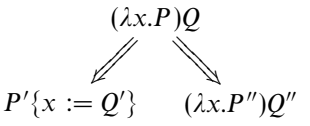


by induction on the derivation of $M \Rightarrow M_1$ (resp. $\theta \Rightarrow \theta_1$). We distinguish the following cases:

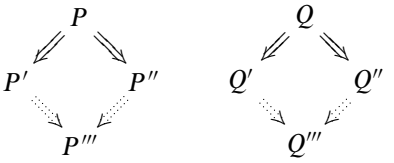
1. (pRef) was applied, with $M = M_1$, take $M_3 = M_2$.
2. (pAppLam) was applied, with $M = (\lambda x.P)Q$, $M_1 = P'\{x := Q'\}$ with $P \Rightarrow P'$, $Q \Rightarrow Q'$. From the definition of $\Rightarrow$, one of the following cases hold:

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

– either $M_2 = (\lambda x.P'')Q''$ with $P \Rightarrow P''$, $Q \Rightarrow Q''$, in which case

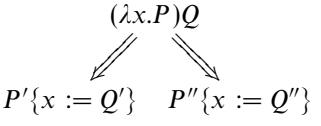


By IH the following diagrams close for some P''', Q''':



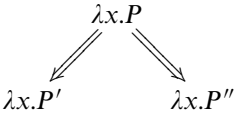
By Proposition 3 (3), the desired diagram is closed taking $M_3 = P'''\{x := Q'''\}$.

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

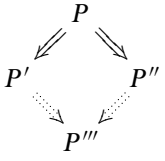


Again by Proposition 3 (3), the diagram is closed taking $M_3 = P'''\{x := Q'''\}$.

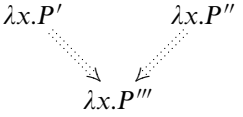
3. (pLam) was applied, with $M = \lambda x.P$, $M_1 = \lambda x.P'$, $P \Rightarrow P''$, and from the definition of $\Rightarrow$, we have $M_2 = \lambda x.P'$ with $P \Rightarrow P'$, in which case



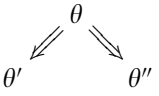
By IH we have



so that

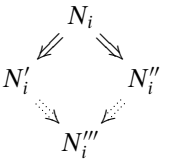


4. (pCBind) was applied, with $\theta = (c_i \mapsto N_i)_{i=1,\ldots,n}$, $\theta' = (c_i \mapsto N_i')_{i=1,\ldots,n}$ with $N_i \Rightarrow N_i'$ for $1 \leqslant i \leqslant n$ and from the definition of $\Rightarrow$ we have $\theta'' = (c_i \mapsto N_i'')_{i=1,\ldots,n}$ with $N_i \Rightarrow N_i''$ for $1 \leqslant i \leqslant n$, in which case

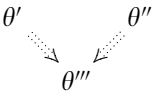


https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

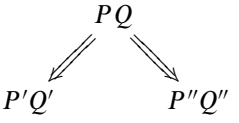
so for every $1 \leqslant i \leqslant n$ we have by IH the diagram



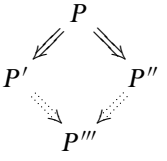
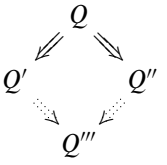
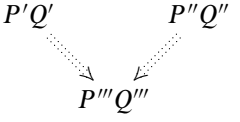
then taking $\theta''' = (c_i \mapsto N_i''')_{i=1,\ldots,n}$



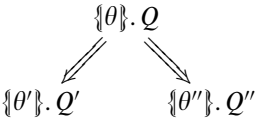
5. (pApp) was applied, with $M = PQ$, $M_1 = P'Q'$ with $P \Rightarrow P'$, $Q \Rightarrow Q'$, so that from the definition of $\Rightarrow$ one of the following cases holds:
   - either $M_2 = P''Q''$ with $P \Rightarrow P''$, $Q \Rightarrow Q''$, in which case



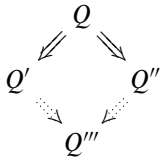
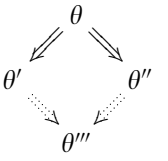
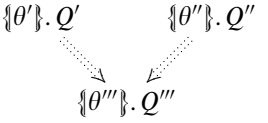
     and by IH we have the diagrams


and

so that


   - or $M_2 = N''\{x := Q''\}$ with $P = \lambda x.N$, $N \Rightarrow N''$ and $Q \Rightarrow Q''$, in which case from the definition of $\Rightarrow$ we know that $P' = \lambda x.N'$ 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 that we have that $M_2 = \{\!|\theta''|\!\}.Q''$ with $\theta \Rightarrow \theta''$, $Q \Rightarrow Q''$ and



and by IH we have


and

so that


□

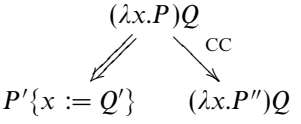
https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

Proof of Lemma 11 p. 598 (Strong commutation of CaseCase *with* $\Rightarrow$*).*
We prove that for all terms M, M_1, M_2 and for all case bindings θ, θ_1, θ_2

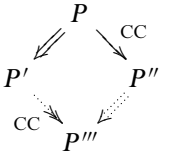
1. If $M \Rightarrow M_1$ and $M \to_{\mathrm{CC}} M_2$, then there exists M_3 such that $M_1 \overset{*}{\to}_{\mathrm{CC}} M_3$ and $M_2 \Rightarrow M_3$.
2. If $\theta \Rightarrow \theta_1$ and $\theta \to_{\mathrm{CC}} \theta_2$, then there exists θ_3 such that $\theta_1 \overset{*}{\to}_{\mathrm{CC}} \theta_3$ and $\theta_2 \Rightarrow \theta_3$.

We reason by mutual induction on the derivations $M \Rightarrow M_1$ and $\theta \Rightarrow \theta_1$. We have the following cases:

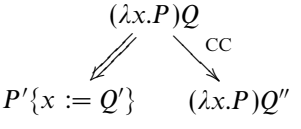
1. (pRef) was applied, with $M = M_1$, take $M_3 = M_2$.
2. (pAppLam) was applied, with $M = (\lambda x.\, P)Q$, $M_1 = P'\{x := Q'\}$, $P \Rightarrow P'$, $Q \Rightarrow Q'$. There are two possibilities for the CC-reduction step:
   - either $M_2 = (\lambda x.P'')Q$ with $P \to_{\mathrm{CC}} P''$, that is



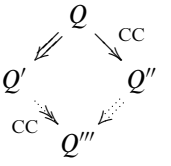
     By IH the following diagram holds for some P''':



     By Corollary 1, $P'\{x := Q'\} \overset{*}{\to}_{\mathrm{CC}} P'''\{x := Q'\}$, 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 $M_2 = (\lambda x.P)Q''$ with $Q \to_{\mathrm{CC}} Q''$, in which case



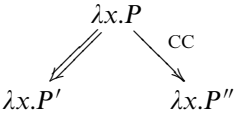
     By IH the following diagram holds for some Q''':



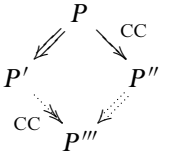
     By Corollary 1, $P'\{x := Q'\} \overset{*}{\to}_{\mathrm{CC}} P'\{x := Q'''\}$, and since $P \Rightarrow P'$ and $Q'' \Rightarrow Q'''$, $(\lambda x.P)Q'' \Rightarrow P'\{x := Q'''\}$ so the diagram is closed taking $M_3 = P'\{x := Q'''\}$.

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

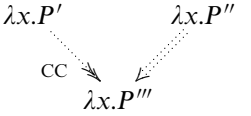
3. (pLam) was applied, with $M = \lambda x.P$, $M_1 = \lambda x.P'$ with $P \Rightarrow P'$, in which case there exists P'' such that



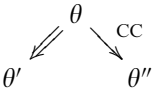
By IH we have



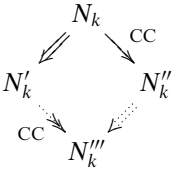
so that



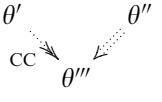
4. (pCBind) was applied, with $\theta = (c_i \mapsto N_i)_{i=1,\ldots,n}$, $\theta' = (c_i \mapsto N'_i)_{i=1,\ldots,n}$, $N_i \Rightarrow N'_i$ for $1 \leqslant i \leqslant n$, in which case



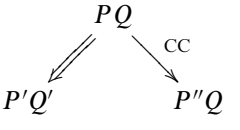
so for some $k \in \{1; \ldots; n\}$ we have by IH the diagram



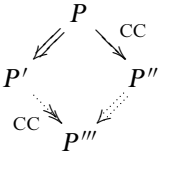
then taking $\theta''' = (c_i \mapsto N'''_i)_{i=1,\ldots,n}$ with $N'''_i = N'_i$ for $i \neq k$



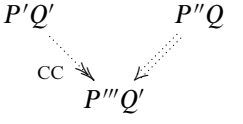
5. (pApp) was applied, with $M = PQ$ and $M_1 = P'Q'$. There are two possibilities for the CC-reduction step:
   - either $M_2 = P''Q$ with $P \to_{\mathrm{CC}} P''$, in which case



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



so that

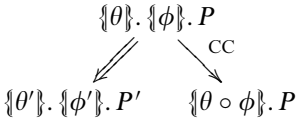


https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

– or $M_2 = PQ''$ with $Q \to_{\mathrm{CC}} Q''$, in which case the diagram is closed analogously.

6. (pCase) was applied, with $M = \{\!|\theta|\!\}.Q$, $M_1 = \{\!|\theta'|\!\}.Q'$, $\theta \Rightarrow \theta'$ and $Q \Rightarrow Q'$. There are two possibilities for the CC-reduction step:

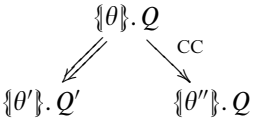
– either CaseCase was applied at the root, i.e. $Q = \{\!|\phi|\!\}.P$, so that $Q' = \{\!|\phi'|\!\}.P'$ with $\phi \Rightarrow \phi'$, $P \Rightarrow P'$ and we have



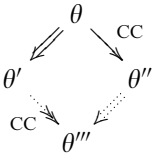
By Prop. 3 (4) we have $\theta \circ \phi \Rightarrow \theta' \circ \phi'$, hence $\{\!|\theta \circ \phi|\!\}.P \Rightarrow \{\!|\theta' \circ \phi'|\!\}.P'$. Since $\{\!|\theta'|\!\}.\{\!|\phi'|\!\}.P' \to_{\mathrm{CC}} \{\!|\theta' \circ \phi'|\!\}.P'$, the diagram is closed.

– or an internal CaseCase was applied, then

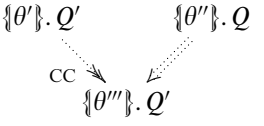
- either $\theta \to_{\mathrm{CC}} \theta''$, so we have



and by IH we have

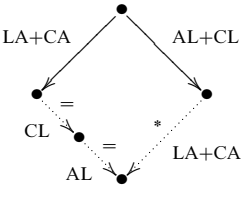


so that

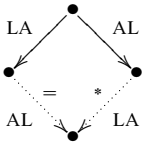


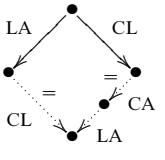
- or $Q \to_{\mathrm{CC}} Q''$, and the diagram is closed analogously. □

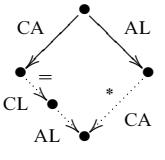
Proof of Lemma 22 p. 602.
The commutation of the first diagram

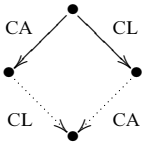


is obtained by merging the following diagrams that cover all the possible cases:



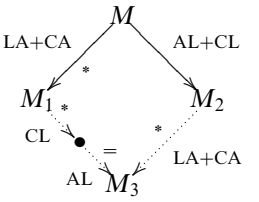






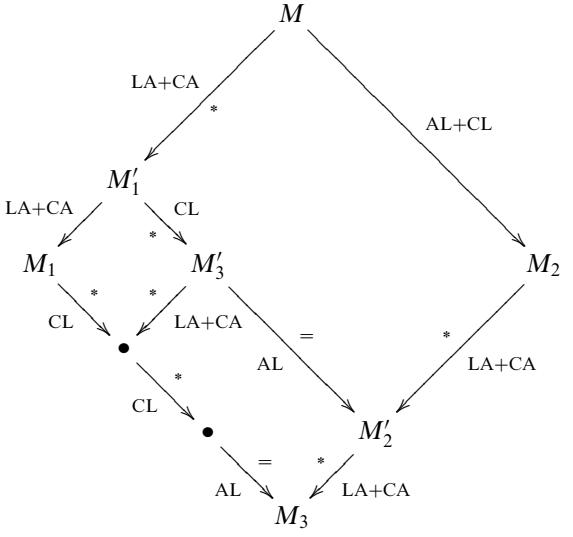
https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

(The diagrams above come from lemmas 10, 20, 18 and 17, respectively.) The commutation of the second diagram



is deduced from the first diagram, by induction on the number of (LamApp+CaseApp)-reduction steps. We distinguish the following cases:

- If the derivation $M \xrightarrow{*}_{\mathrm{LA+CA}} M_1$ has 0 steps, the result is obvious.
- If it has 1 step, this is the first diagram (see above).
- So let us assume it has $\geqslant 2$ steps. We proceed by lexicographic induction on the pair (LA + CA + CL-depth of M, length of the LA + CA-derivation). The picture is



The upper right rectangle is closed by IH. The left-most square is closed since LA + CA (strongly) commutes with CL. At M_3' the lexicographic pair clearly has a value which is less than the value at M, since the length of $M \xrightarrow{*}_{\mathrm{LA+CA}} M_1'$ is $\geqslant 1$, so IH allows to close the lower rectangle. □

Proof of Proposition 13 p. 612 (Separation of disagreeing terms).
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:
   - $M_1 = \maltese$ and $M_2 = \lambda x_1 \cdots x_n.HN_1 \cdots N_k$.
     In this case, take a closed evaluation context $E[]$ such that $E\left[M_2[\sigma_X^K]\right]$ is undefined (by Lemma 27) and conclude using Lemma 24.

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

– $M_1 = \lambda x_1 \cdots x_n . H N_1 \cdots N_k$ and $M_2 = \maltese$.
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 (Definition 13) at the begining of Section 5.5:

- $H_1 = y$ (where y is a variable), and $H_2 = c$ (where c is a constructor). Take:

$$E[] \equiv \{\!|c \mapsto \maltese|\!\}.\, [] \langle \mathtt{x}_1 ; *_K \rangle \cdots \langle \mathtt{x}_n ; *_K \rangle ?_{k_1+1} \cdots ?_K U$$

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

$$\begin{array}{rcl} E\left[M_1[\sigma_X^K]\right] & = & \left(\{\!|c \mapsto \maltese|\!\}.\, (\lambda x_1 \cdots x_n . y N_{1,1} \cdots N_{1,k_1})[\sigma_X^K]\right) \\ & & \quad \langle \mathtt{x}_1 ; *_K \rangle \cdots \langle \mathtt{x}_n ; *_K \rangle ?_{k_1+1} \cdots ?_K U \\ & \to^* & (\lambda x_1 \cdots x_n . (\{\!|c \mapsto \maltese|\!\}.\, y) N_{1,1} \cdots N_{1,k_1})[\sigma_X^K] \\ & & \quad \langle \mathtt{x}_1 ; *_K \rangle \cdots \langle \mathtt{x}_n ; *_K \rangle ?_{k_1+1} \cdots ?_K U \\ & \to^* & (\{\!|c \mapsto \maltese|\!\}.\, \langle \mathtt{y} ; *_K \rangle) \\ & & \quad N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] ?_{k_1+1} \cdots ?_K U \\ & \to^* & \{\!|c \mapsto \maltese \mid \mathtt{y} ; *_K |\!\}\, N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] ?_{k_1+1} \cdots ?_K U \\ & \to^* & (\{\!|c \mapsto \maltese|\!\}.\, U) \mathtt{y} N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] ?_{k_1+1} \cdots ?_K, \end{array}$$

the latter term being undefined by Lemma 25, whereas

$$\begin{array}{rcl} E\left[M_2[\sigma_X^K]\right] & = & \left(\{\!|c \mapsto \maltese|\!\}.\, (\lambda x_1 \cdots x_n . c N_{2,1} \cdots N_{2,k_2})[\sigma_X^K]\right) \\ & & \quad \langle \mathtt{x}_1 ; *_K \rangle \cdots \langle \mathtt{x}_n ; *_K \rangle ?_{k_1+1} \cdots ?_K U \\ & \to^* & (\lambda x_1 \cdots x_n . (\{\!|c \mapsto \maltese|\!\}.\, c) N_{2,1} \cdots N_{2,k_2})[\sigma_X^K] \\ & & \quad \langle \mathtt{x}_1 ; *_K \rangle \cdots \langle \mathtt{x}_n ; *_K \rangle ?_{k_1+1} \cdots ?_K U \\ & \to^* & (\lambda x_1 \cdots x_n . \maltese N_{2,1} \cdots N_{2,k_2})[\sigma_X^K] \\ & & \quad \langle \mathtt{x}_1 ; *_K \rangle \cdots \langle \mathtt{x}_n ; *_K \rangle ?_{k_1+1} \cdots ?_K U \\ & \to^* & \maltese \end{array}$$

The symmetric case is treated the same way.

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

$$E[] \equiv \{\!|c \mapsto \maltese|\!\}.\, [] \langle \mathtt{x}_1 ; *_K \rangle \cdots \langle \mathtt{x}_n ; *_K \rangle ?_{k_1+1} \cdots ?_K U$$

where $?_{k_1+1}, \ldots, ?_K$ are arbitrary closed terms, and where U is an arbitrary closed undefined head term. As in the latter case, we easily check that $E\left[M_1[\sigma_X^K]\right]$ is undefined whereas $E\left[M_2[\sigma_X^K]\right] \to^* \maltese$. (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.

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

$$E[] \equiv \{\!|c \mapsto \maltese|\!\}.\, [] \langle \mathtt{x}_1 ; *_K \rangle \cdots \langle \mathtt{x}_n ; *_K \rangle \underbrace{c \cdots c}_{K+1 \text{ times}}$$

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

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

$$\begin{array}{rcl}
E\big[M_1[\sigma_X^K]\big] & = & \big(\{\!|c \mapsto \maltese|\!\}.(\lambda x_1 \cdots x_n.\, \{\!|\theta|\!\}.\, y_1 N_{1,1} \cdots N_{1,k_1})[\sigma_X^K]\big) \\
& & \quad \langle \mathtt{x}_1 ; *_K \rangle \cdots \langle \mathtt{x}_n ; *_K \rangle \underbrace{c \cdots c}_{K+1} \\
& \to^* & (\lambda x_1 \cdots x_n.\, (\{\!|(c \mapsto \maltese) \circ \theta|\!\}.\, y_1) N_{1,1} \cdots N_{1,k_1})[\sigma_X^K] \\
& & \quad \langle \mathtt{x}_1 ; *_K \rangle \cdots \langle \mathtt{x}_n ; *_K \rangle \underbrace{c \cdots c}_{K+1} \\
& \to^* & (\{\!|(c \mapsto \maltese) \circ \theta[\sigma_{X'}^K]|\!\}.\, \langle \mathtt{y}_1 ; *_K \rangle) \\
& & \quad N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] \underbrace{c \cdots c}_{K+1} \\
& \to^* & \{\!|(c \mapsto \maltese) \circ \theta[\sigma_{X'}^K] \mid \mathtt{y}_1 ; *_K |\!\}\!\rangle \\
& & \quad N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] \underbrace{c \cdots c}_{K+1} \\
& \to^* & (\{\!|(c \mapsto \maltese) \circ \theta[\sigma_{X'}^K]|\!\}.\, c) \\
& & \quad \mathtt{y}_1 N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] \underbrace{c \cdots c}_{K}
\end{array}$$

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

$$\begin{array}{rcl}
E\big[M_2[\sigma_X^K]\big] & \to^* & (\{\!|c \mapsto \maltese|\!\}.\, c)\ \mathtt{y}_2 N_{2,1}[\sigma_{X'}^K] \cdots N_{2,k_2}[\sigma_{X'}^K] \underbrace{c \cdots c}_{K} \\
& \to^* & \maltese\ \mathtt{y}_2\ N_{2,1}[\sigma_{X'}^K] \cdots N_{2,k_2}[\sigma_{X'}^K] \underbrace{c \cdots c}_{K} \\
& \to^* & \maltese
\end{array}$$

The symmetric case is treated the same way.

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

$$\begin{array}{rcl}
P & \equiv & \lambda z.\, \mathsf{if}\ (\mathsf{eq}\ z\ \mathtt{y}_1)\ \maltese\ U \\
E[] & \equiv & []\langle \mathtt{x}_1 ; *_K \rangle \cdots \langle \mathtt{x}_n ; *_K \rangle \underbrace{P \cdots P}_{K+1 \text{ times}}
\end{array}$$

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

$$\begin{array}{rcl}
E\big[M_1[\sigma_X^K]\big] & = & (\lambda x_1 \cdots x_n.\, y_1 N_{1,1} \cdots N_{1,k_1})[\sigma_X^K] \\
& & \quad \langle \mathtt{x}_1 ; *_K \rangle \cdots \langle \mathtt{x}_n ; *_K \rangle \underbrace{P \cdots P}_{K+1} \\
& \to^* & \langle \mathtt{y}_1 ; *_K \rangle N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] \underbrace{P \cdots P}_{K+1} \\
& \to^* & P\ \mathtt{y}_1 N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] \underbrace{P \cdots P}_{K} \\
& \to^* & \maltese\ N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] \underbrace{P \cdots P}_{K} \\
& \to^* & \maltese
\end{array}$$

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

On the other hand we have

$$\begin{aligned} E\big[M_2[\sigma_X^K]\big] &\to^* P\,\mathtt{y}_2 N_{2,1}[\sigma_{X'}^K]\cdots N_{2,k_2}[\sigma_{X'}^K]\underbrace{P\cdots P}_{K} \\ &\to^* U N_{2,1}[\sigma_{X'}^K]\cdots N_{2,k_2}[\sigma_{X'}^K]\underbrace{P\cdots P}_{K} \end{aligned}$$

which is undefined by Lemma 25.

- $H_1 = c_1$ and $H_2 = c_2$ (where c_1 and c_2 are constructors), but $c_1 \neq c_2$. Take

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

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

$$\begin{aligned} E\big[M_1[\sigma_X^K]\big] &= \{\!|\theta|\!\}.(\lambda x_1\cdots x_n.\, c_1 N_{1,1}\cdots N_{1,k_1})[\sigma_X^K] \\ &\to^* (\lambda x_1\cdots x_n.(\{\!|\theta|\!\}.\,c_1)N_{1,1}\cdots N_{1,k_1})[\sigma_X^K] \\ &\to^* (\lambda x_1\cdots x_n.\maltese N_{1,1}\cdots N_{1,k_1})[\sigma_X^K] \\ &\to^* \maltese \end{aligned}$$

whereas

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

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 $y_1 \neq y_2$. This case is treated similarly to the case where H_1 and H_2 are distinct variables, by setting

$$\begin{aligned} P &\equiv \lambda z.\,\mathsf{if}\ (\mathsf{eq}\ z\ \mathtt{y}_1)\ \maltese\ U \\ E[] &\equiv []\langle \mathtt{x}_1; *_K\rangle\cdots\langle \mathtt{x}_n; *_K\rangle\underbrace{P\cdots P}_{K+1\ \text{times}} \end{aligned}$$

(where U is an arbitrary closed 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 $\mathrm{dom}(\theta_1) \neq \mathrm{dom}(\theta_2)$. Without loss of generality, assume that c_1 is a constructor such that $c_1 \in \mathrm{dom}(\theta_1)$ and $c_1 \notin \mathrm{dom}(\theta_2)$ (the other case is treated by symmetry). Writing $X' = X \cup \{x_1; \ldots; x_n\}$ as usual, let us set:

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

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

We have:

$$\begin{array}{rcl}
E_0\big[M_1[\sigma^K_X]\big] & = & (\lambda x_1 \dots x_n.\, \{\!|\theta_1|\!\}.\, y N_{1,1} \cdots N_{1,k_1})[\sigma^K_X] \\
& & \quad \langle \mathtt{x}_1 ; *_K \rangle \cdots \langle \mathtt{x}_n ; *_K \rangle \underbrace{P \cdots P}_{K+1} \\
& \to^* & (\{\!|\theta_1[\sigma^K_{X'}]|\!\}.\, \langle \mathtt{y} ; *_K \rangle) N_{1,1}[\sigma^K_{X'}] \cdots N_{1,k_1}[\sigma^K_{X'}] \underbrace{P \cdots P}_{K+1} \\
& \to^* & \langle\!\!\langle \theta_1[\sigma^K_{X'}] \mid \mathtt{y} ; *_K \rangle\!\!\rangle N_{1,1}[\sigma^K_{X'}] \cdots N_{1,k_1}[\sigma^K_{X'}] \underbrace{P \cdots P}_{K+1} \\
& \to^* & (\{\!|\theta_1[\sigma^K_{X'}]|\!\}.\, P) \mathtt{y} N_{1,1}[\sigma^K_{X'}] \cdots N_{1,k_1}[\sigma^K_{X'}] \underbrace{P \cdots P}_{K} \\
& \to^* & (\lambda z_0 z_1 \dots z_{2K}.\, \{\!|\theta_1[\sigma^K_{X'}]|\!\}.\, c_1) \\
& & \quad \mathtt{y} N_{1,1}[\sigma^K_{X'}] \cdots N_{1,k_1}[\sigma^K_{X'}] \underbrace{P \cdots P}_{K} \\
& \to^* & \lambda z_{k_1+K+1} \dots z_{2K}.\, \{\!|\theta_1[\sigma^K_{X'}]|\!\}.\, c_1 \\
& \to^* & \lambda z_{k_1+K+1} \dots z_{2K}.\, \theta_1(c_1)[\sigma^K_{X'}]
\end{array}$$

whereas

$$E_0\big[M_2[\sigma^K_X]\big] \quad \to^* \quad \lambda z_{k_2+K+1} \dots z_{2K}.\, \{\!|\theta_2[\sigma^K_{X'}]|\!\}.\, c_1$$

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 27, there exists an evaluation context $F[]$ such that

$$F\big[T[\sigma^K_{X'}]\big] \;=\; F\big[\lambda z_{k_1+K+1} \dots z_{2K}.\, \theta_1(c_1)[\sigma^K_{X'}]\big] \;\to^*\; \maltese$$

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

$$E\big[M_1[\sigma^K_{X'}]\big] \;=\; F\big[E_0\big[M_1[\sigma^K_{X'}]\big]\big] \;\to^*\; F\big[T[\sigma^K_{X'}]\big] \;\to^*\; \maltese$$

whereas

$$E\big[M_2[\sigma^K_{X'}]\big] \;\to^*\; F\big[\lambda z_{k_2+K+1} \dots z_{2K}.\, \{\!|\theta_2[\sigma^K_{X'}]|\!\}.\, c_1\big]$$

which is undefined by Lemma 25.

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 $1 \leqslant k \leqslant \min(k_1, k_2)$ such that $\mathrm{dis}_d(N_{1,k}, N_{2,k})$. By induction hypothesis, there exists a closed evaluation context $E_0[]$ such that

   $$E_0\big[N_{1,k}[\sigma^K_{X'}]\big] \to^* \maltese \qquad \text{and} \qquad E_0\big[N_{2,k}[\sigma^K_{X'}]\big] \;\text{ is undefined}$$

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

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

- $H_1 = H_2 = y$. Let us set

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

We check that

$$\begin{array}{rcl} E\big[M_1[\sigma_X^K]\big] & = & (\lambda x_1 \dots x_n . y N_{1,1} \cdots N_{1,k_1})[\sigma_X^K] \\ & & \quad \langle \mathtt{x}_1 ; *_K \rangle \cdots \langle \mathtt{x}_n ; *_K \rangle \underbrace{P \cdots P}_{K+1} \\ & \to^* & \langle \mathtt{y} ; *_K \rangle N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] \underbrace{P \cdots P}_{K+1} \\ & \to^* & P \mathtt{y} N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] \underbrace{P \cdots P}_{K} \\ & \to^* & E_0\big[N_{1,k}[\sigma_{X'}^K]\big] \underbrace{P \cdots P}_{k_1} \\ & \to^* & \maltese \underbrace{P \cdots P}_{k_1} \\ & \to^* & \maltese \end{array}$$

whereas

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

which is undefined by Lemma 25.

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

$$\begin{array}{rcl} \theta & \equiv & (c \mapsto \lambda z_1 \dots z_k . E_0[z_k]) \\ E[] & \equiv & \{\!|\theta|\!\}.\, []\langle \mathtt{x}_1 ; *_K \rangle \cdots \langle \mathtt{x}_n ; *_K \rangle \end{array}$$

We check that

$$\begin{array}{rcl} E\big[M_1[\sigma_X^K]\big] & = & \{\!|\theta|\!\}.\, (\lambda x_1 \dots x_n . c N_{1,1} \cdots N_{1,k_1})[\sigma_X^K] \\ & & \quad \langle \mathtt{x}_1 ; *_K \rangle \cdots \langle \mathtt{x}_n ; *_K \rangle \\ & \to^* & (\lambda x_1 \dots x_n . (\{\!|\theta|\!\}.\, c) N_{1,1}[\sigma_X^K] \cdots N_{1,k_1}[\sigma_X^K]) \\ & & \quad \langle \mathtt{x}_1 ; *_K \rangle \cdots \langle \mathtt{x}_n ; *_K \rangle \\ & \to^* & (\{\!|\theta|\!\}.\, c) N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] \quad \text{(since } \theta \text{ closed)} \\ & \to^* & E_0\big[N_{1,k}[\sigma_{X'}^K]\big] N_{1,k+1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] \\ & \to^* & \maltese N_{1,k+1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] \\ & \to^* & \maltese \end{array}$$

whereas

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

which is undefined by Lemma 25.

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

– $H_1 = \{\!|\theta_1|\!\}.\, y$ and $H_2 = \{\!|\theta_2|\!\}.\, y$ for some case bindings θ_1, θ_2 and for some variable y, and there is a constructor $c \in \mathrm{dom}(\theta_1) = \mathrm{dom}(\theta_2)$ such that $\mathrm{dis}_d(\theta_1(c), \theta_2(c))$. 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

$$E_0\big[\theta_1(c)[\sigma^K_{X'}]\big] \to^* \maltese \qquad \text{and} \qquad E_0\big[\theta_2(c)[\sigma^K_{X'}]\big] \text{ is undefined}$$

(or conversely). Let us then set

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

We have

$$\begin{aligned} E'\big[M_1[\sigma^K_X]\big] &= (\lambda x_1 \cdots x_n.\, \{\!|\theta_1|\!\}.\, y N_{1,1} \cdots N_{1,k})[\sigma^K_X] \\ &\quad \langle \mathtt{x}_1 ; *_K\rangle \cdots \langle \mathtt{x}_n ; *_K\rangle \underbrace{P \cdots P}_{K+1-k} \\ &\to^* (\{\!|\theta_1[\sigma^K_{X'}]|\!\}.\, \langle \mathtt{y}, *_K\rangle) N_{1,1}[\sigma^K_{X'}] \cdots N_{1,k}[\sigma^K_{X'}] \underbrace{P \cdots P}_{K+1-k} \\ &\to^* \{\!|\theta_1[\sigma^K_{X'}] \mid \mathtt{y}, *_K|\!\} N_{1,1}[\sigma^K_{X'}] \cdots N_{1,k}[\sigma^K_{X'}] \underbrace{P \cdots P}_{K+1-k} \\ &\to^* (\{\!|\theta_1[\sigma^K_{X'}]|\!\}.\, P) \mathtt{y} N_{1,1}[\sigma^K_{X'}] \cdots N_{1,k}[\sigma^K_{X'}] \underbrace{P \cdots P}_{K-k} \\ &\to^* (\lambda z_0 z_1 \dots z_K.\, \{\!|\theta_1[\sigma^K_{X'}]|\!\}.\, c) \\ &\quad \mathtt{y} N_{1,1}[\sigma^K_{X'}] \cdots N_{1,k}[\sigma^K_{X'}] \underbrace{P \cdots P}_{K-k} \\ &\to^* \{\!|\theta_1[\sigma^K_{X'}]|\!\}.\, c \;\to\; \theta_1(c)[\sigma^K_{X'}] \end{aligned}$$

Similarly, we have

$$E'\big[M_2[\sigma^K_X]\big] \;\to^*\; \{\!|\theta_2[\sigma^K_{X'}]|\!\}.\, c \;\to\; \theta_2(c)[\sigma^K_{X'}].$$

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

$$E\big[M_1[\sigma^K_X]\big] \;=\; E_0\big[E'\big[M_1[\sigma^K_X]\big]\big] \;\to^*\; E_0\big[\theta_1(c)[\sigma^K_{X'}]\big] \;\to^*\; \maltese$$

whereas

$$E\big[M_2[\sigma^K_X]\big] \;=\; E_0\big[E'\big[M_2[\sigma^K_X]\big]\big] \;\to^*\; E_0\big[\theta_2(c)[\sigma^K_{X'}]\big],$$

which is undefined.

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

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

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

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

$$\begin{aligned}
E\big[M_1[\sigma_X^K]\big] &= (\lambda x_1 \cdots x_n.\, \{\!\{\theta_1\}\!\}.\, y N_{1,1} \cdots N_{1,k_1})[\sigma_X^K] \\
&\quad \langle \mathtt{x}_1 ; *_K \rangle \cdots \langle \mathtt{x}_n ; *_K \rangle \underbrace{? \cdots ?}_{K-k_2}\, U\, \underbrace{? \cdots ?}_{k_2-k_1-1} \Lightning \\
&\to^* (\{\!\{\theta_1[\sigma_{X'}^K]\}\!\}.\, \langle \mathtt{y} ; *_K \rangle) \\
&\quad N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] \underbrace{? \cdots ?}_{K-k_2}\, U\, \underbrace{? \cdots ?}_{k_2-k_1-1} \Lightning \\
&\to^* \langle\!\langle \theta_1[\sigma_{X'}^K] \mid \mathtt{y} ; *_K \rangle\!\rangle \\
&\quad N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] \underbrace{? \cdots ?}_{K-k_2}\, U\, \underbrace{? \cdots ?}_{k_2-k_1-1} \Lightning \\
&\to^* \{\!\{\theta_1[\sigma_{X'}^K]\}\!\}.\, \Lightning \mathtt{y} N_{1,1}[\sigma_{X'}^K] \cdots N_{1,k_1}[\sigma_{X'}^K] \underbrace{? \cdots ?}_{K-k_2}\, U\, \underbrace{? \cdots ?}_{k_2-k_1-1} \\
&\to^* \Lightning
\end{aligned}$$

whereas

$$\begin{aligned}
E\big[M_2[\sigma_X^K]\big] &\to^* \langle\!\langle \theta_2[\sigma_{X'}^K] \mid \mathtt{y} ; *_K \rangle\!\rangle \\
&\quad N_{2,1}[\sigma_{X'}^K] \cdots N_{2,k_2}[\sigma_{X'}^K] \underbrace{? \cdots ?}_{K-k_2}\, U\, \underbrace{? \cdots ?}_{k_2-k_1-1} \Lightning \\
&\to^* \{\!\{\theta_2[\sigma_{X'}^K]\}\!\}.\, U \mathtt{y} N_{2,1}[\sigma_{X'}^K] \cdots N_{2,k_2}[\sigma_{X'}^K] \underbrace{? \cdots ?}_{K-k_2} \underbrace{? \cdots ?}_{k_2-k_1-1} \Lightning
\end{aligned}$$

which is undefined. □

Appendix B. Confluence of the whole system $\lambda\mathscr{B}_{\mathscr{C}}$

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 $(s_1, s_2) \models$ BCC and $(s_1 + s_2) \models$ 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. (AL $\models$ CR) [Table 1 (1)]
2. (AL // AD) [Table 1 (2)]
3. (AL // CO) [Table 1 (4)]
4. (AL // CD) [Table 1 (5)]
5. (AL // CL) [Table 1 (6)]
6. (AL // CD + CL) since (AL // CD) [4.] and (AL // CL) [5.]
7. (AL // AD + CD + CL) since (AL // AD) [2.] and (AL // CD + CL) [6.]
8. (AL // AL + AD + CD + CL) since (AL $\models$ CR) [1.] and (AL // AD + CD + CL) [7.]
9. (AL // CC) [Table 1 (7)]
10. (LA + LD + CD + CA // AL + AD + CD + CL) [Table 1 (12)]
11. (AL // CL + CC) since (AL // CL) [5.] and (AL // CC) [9.]
12. (AL // CD + CL + CC) since (AL // CD) [4.] and (AL // CL + CC) [11.]
13. (AL // CO + CD + CL + CC) since (AL // CO) [3.] and (AL // CD + CL + CC) [12.]
14. (AL // AD + CO + CD + CL + CC) since (AL // AD) [2.] and (AL // CO + CD + CL + CC) [13.]

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press

15. (AD + CD + CL // AD + CO + CD + CL + CC) since BCC + SN
16. (AL + AD + CD + CL // AD + CO + CD + CL + CC) since
    (AL // AD + CO + CD + CL + CC) [14.] and
    (AD + CD + CL // AD + CO + CD + CL + CC) [15.]
17. (AL + AD + CD + CL // AD + LA + LD + CO + CD + CA + CL + CC) since
    (LA + LD + CD + CA // AL + AD + CD + CL) [10.] and
    (AL + AD + CD + CL // AD + CO + CD + CL + CC) [16.]
18. (AL + AD + CD + CL // AL + AD + LA + LD + CO + CD + CA + CL + CC) since
    (AL // AL + AD + CD + CL) [8.] and
    (AL + AD + CD + CL // AD + LA + LD + CO + CD + CA + CL + CC) [17.]
19. (LA + LD + CO + CD + CA + CL + CC // AD + LA + LD + CO + CD + CA + CL + CC)
    since BCC + SN
20. (AD + LA + LD + CO + CD + CA + CL + CC //
    AL + AD + LA + LD + CO + CD + CA + CL + CC) since
    (AL + AD + CD + CL // AD + LA + LD + CO + CD + CA + CL + CC) [17.] and
    (LA + LD + CO + CD + CA + CL + CC // AD + LA + LD + CO + CD + CA + CL + CC) [19.]
21. (AL + AD + LA + LD + CO + CD + CA + CL + CC ⊨ CR) since
    (AL + AD + CD + CL // AL + AD + LA + LD + CO + CD + CA + CL + CC) [18.] and
    (AD + LA + LD + CO + CD + CA + CL + CC //
    AL + AD + LA + LD + CO + CD + CA + CL + CC) [20.]

https://doi.org/10.1017/S0956796809007369 Published online by Cambridge University Press