Call-by-Value Lambda-calculus and LJQ

Roy Dyckhoff, Stéphane Lengrand

Call-by-Value λ-calculus and LJQ

Roy Dyckhoff and Stéphane Lengrand
School of Computer Science,
University of St Andrews,
St Andrews, Fife,
KY16 9SX, Scotland.
{rd,sl}@dcs.st-and.ac.uk

19th April 2007

Abstract

LJQ is a focused sequent calculus for intuitionistic logic, with a simple restriction on the first premiss of the usual left introduction rule for implication. In a previous paper we discussed its history (going back to about 1950, or beyond) and presented its basic theory and some applications; here we discuss in detail its relation to call-by-value reduction in lambda calculus, establishing a connection between LJQ and the CBV calculus λ_{C} of Moggi. In particular, we present an equational correspondence between these two calculi forming a bijection between the two sets of normal terms, and allowing reductions in each to be simulated by reductions in the other.

Keywords: lambda calculus, call-by-value, sequent calculus, continuation-passing, semantics

1 Introduction

Proof systems for intuitionistic logic are well-known to be related to computation. For example, a system of uniform proofs for Horn logic is a coherent explanation of proof search in pure Prolog, as argued by (e.g.) [17]; likewise, natural deduction corresponds under the Curry-Howard correspondence to the ordinary typed λ-calculus, with beta-reduction, i.e. to the classic model of computation for typed functional programs.

Similarly, the sequent calculus can also be related to computation via the process of *cut-elimination*. In particular, two intuitionistic sequent calculi LJT and LJQ, presented in Herbelin's thesis [10] and based on earlier work [14], [2] by Joinet *et al*, feature cut-elimination systems in connection with call-by-name (CBN) and call-by-value (CBV) semantics, respectively.

Such semantics apply to functional programming [20] and concern the choice of evaluating an argument after or (resp.) before being passed to a function.

The λ-calculus is so pure a model for functional programs that *any* computation is, in a sense defined in [20], call-by-name. This relates to the sequent calculus LJT whose notion of *focus* [1] establishes a close connection with natural deduction, of which the λ-calculus is a proof-term language. The cut-free proofs of LJT are in 1-1 correspondence with the normal natural deductions; LJT has a well-developed theory of proof-reduction with strong normalisation [11, 5, 3] that relates to the normalisation of the simply-typed λ-calculus. Note that LJT itself can be given with proof-terms, which help formalising such connections directly with call-by-name (i.e. unrestricted) λ-calculus; these terms are meaningful even in an untyped setting.

Likewise, LJQ (as presented in [4]) is a focused sequent calculus, capturing a simple restriction on proofs significant for structural proof theory and automated reasoning. In [10], LJQ is related to Lorenzen dialogues, but the suggested connection with call-by-value semantics has never, as far as we are aware, been formalised as a relation between cut-elimination in LJQ and normalisation in a call-by-value (λ-)calculus.

Addressing this issue raises the questions of what call-by-value λ-calculus is, and what cut-elimination process specific to LJQ is to be used. These questions have very satisfying answers for CBN and LJT(c.f. [11, 5, 3]); extending (for implicational logic) the brief discussion in [4], the present paper

- formalises, by means of a proof-term calculus, a cut-elimination system specific to LJQ and its particular notion of focus,
- optimises a CBV λ-calculus due to Moggi [18], so that it satisfies finer semantical properties and thus better connects to LJQ,
- proves the confluence of the obtained calculi and relates them by using notions of abstract rewriting such as *equational equivalences*.

The logical inspiration for this investigation is strong, but once proof-terms are considered, all results in this paper hold even in an untyped framework.

Call-by-value λ-calculus

Recall that CBV λ-calculus partially (and only partially) captures the common convention, in implementation of functional languages, that arguments are evaluated *before* being passed to functions. The fundamental ideas are explored by Plotkin in [20], where a calculus λ_{V} is introduced; its terms are exactly those of λ-calculus and its reduction rule β_{V} is merely β-reduction restricted to the case where the argument is a *value* V, i.e. a variable or an abstraction.

$$\beta_{\mathsf{V}} \qquad (\lambda x.M)\, V \longrightarrow M\{x = V\}$$

Clearly, in the fragment $\lambda_{\mathsf{EvalArg}}$ of λ-calculus where all arguments are values, $\beta_{\mathsf{V}} = \beta$. Hence, applying CBN- (general β) or CBV- (β_{V}) reduction is the same, a property that is called *strategy indifference*; therefore, in the context of $\lambda_{\mathsf{EvalArg}}$, we will not distinguish β_{V} from β.

Using the notion of *continuation*, of which a history can be found in [22], the (CBN) λ-calculus can be encoded [20] into $\lambda_{\mathsf{EvalArg}}$ with a *continuation-passing-style* (*CPS*) translation in a way such that two terms are β-equivalent if and only if their encodings are β-equivalent.

Similarly, for the CBV calculus λ_{V}, two CPS-translations into $\lambda_{\mathsf{EvalArg}}$ can be defined: Reynolds' [21, 23] and Fischer's [7, 8]. Both are *sound*, in that in each case any two β_{V}-equivalent λ-terms are mapped to two β-equivalent terms, but they are both *incomplete* in that the β-equivalence in $\lambda_{\mathsf{EvalArg}}$ is bigger than needed (see e.g. [20]).

This problem is solved by Moggi's calculus λ_{C} [18]; this extends λ_{V} with a $\mathsf{let}\ \_ = \_\ \mathsf{in}\ \_$ constructor and new reduction rules, such as:

$$\mathsf{let}_{\mathsf{V}} \qquad \mathsf{let}\ x = V\ \mathsf{in}\ M \longrightarrow M\{x = V\}$$

In particular, with the use of this new constructor, an application cannot be a normal form if one of its components is not a value. Both the aforementioned CPS-translations on λ_{V} can be extended to λ_{C}, in a way such that they are both sound and complete.

Thus, equivalence in the extended CBV λ-calculus λ_{C} of Moggi can be modelled by equivalence in $\lambda_{\mathsf{EvalArg}}$; but (as we show below) this modelling does not extend to modelling of reductions. Our more refined analysis, presented in this paper, of the CBV λ-calculus λ_{C} considers how reductions, rather than equivalence, are modelled by CPS-translations. Here the crucial concept is that of a *reflection*, i.e. a reduction-preserving encoding that makes one calculus isomorphic to a sub-calculus of another. Already in [25], a refined

Reynolds translation is proved to form a *reflection* in λ_{C} of its target calculus $\lambda^{\mathcal{R}}_{\mathsf{CPS}}$.

However, [25] states that a particular refined Fischer translation cannot be a reflection. We show here that a different (and more natural) refined version of the Fischer translation that *does* form a reflection of its target calculus $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$ in (a minor variation of) λ_{C}.

We leave the discussion about the different refinements for section 3.1. The minor variation of λ_{C} consists of the replacement of rule β_{V} with the following:

$$\mathsf{B} \qquad (\lambda x.M)\ N \longrightarrow \mathsf{let}\ x = N\ \mathsf{in}\ M$$

This change is justified for three reasons:

- It does not change the equational theory of λ_{C}.
- Splitting β_{V} into B followed by $\mathsf{let}_{\mathsf{V}}$ seems more atomic and natural, with only one rule controlling whether a particular sub-term is a value or not.
- This variation makes our refined Fischer translation a reflection in λ_{C} of its target calculus.

From the reflection result we can also prove that our version of λ_{C} is confluent (Theorem 1), using the confluence of $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$.

In summary, then, we are able to show that a minor variation of Moggi's calculus λ_{C} is a natural candidate as the canonical CBV λ-calculus, admitting two CPS-translations each of which not merely models equivalences but also models reductions. We turn now to using this as a basis for considering the properties of LJQ -as a CBV sequent calculus (analogous to the use of CBN λ-calculus in relation to the properties of LJT as a CBN sequent calculus, viz. strong normalisation, confluence and simulation of β-reduction).

LJQ

The history of LJQ goes back to Vorob'ev [26] (detailing ideas published in 1952), who showed (his Theorem 3) that, in a minor variant of Gentzen's calculus LJ for intuitionistic logic, one may, without losing completeness, restrict instances of the left rule $L\supset$ for implication to those in which, if the antecedent A of the principal formula $A\supset B$ is atomic, then the first premiss is an axiom. Much later, Hudelmaier [12] showed that one could further restrict this rule to those instances where the first premiss was either an axiom or the conclusion of a right rule; the result was proved in his thesis [12] and described in [13] as folklore. The same result is mentioned by Herbelin in [10] as the completeness of a certain calculus LJQ, described simply as LJ with the last-mentioned restriction.

Following our approach in [4], we formalise such a restriction on LJ as a focussed calculus with two forms of sequent, *ordinary sequents* and *focussed sequents*. We also use proof-terms, so that LJQ becomes the typing system of a calculus called λ_{Q} with two syntactic categories, that of *(ordinary) terms* and that of *values*. These are respectively typed by ordinary sequents and focussed sequents.

Letting Γ range over environments (i.e. finite functions from variables to formulae), we have the *ordinary* sequent $\Gamma \Rightarrow M : A$ to express the deducibility (as witnessed by the term M) of the formula A from the assumptions Γ, and the *focussed* sequent $\Gamma \rightarrow M : A$ to impose the restriction that the last step in the deduction is by an axiom or a right rule (i.e. with the succedent formula principal).

The rules of the calculus are then as presented below, in Sect. 4. For example, the rule $L\supset$ has, as conclusion and second premiss, ordinary sequents, but as first premiss a focussed sequent, capturing the restriction on proofs discussed by Hudelmaier (given that the focussed sequents are exactly the axioms or the conclusions of right introduction rules).

λ_{Q} can be considered independently from typing, while its notion of reduction (not necessarily representing proof transformations) still expresses the call-by-value operational semantics of the calculus. This is revealed by *continuation-passing style* translations [22] and relates to Moggi's call-by-value calculus λ_{C} [18], modified as outlined above.

The connection between λ_{Q} and λ_{C} can be summarised as follows: There are encodings from λ_{Q} to λ_{C} and from λ_{C} to λ_{Q} such that

- they form a bijection between cut-free terms (a.k.a. normal forms) of λ_{Q} and normal forms of λ_{C},
- they form an equational correspondence between λ_{Q} and λ_{C},
- the reductions in λ_{Q} are simulated by those of λ_{C},
- the reductions in λ_{C} are simulated by those of λ_{Q}.

Thus, LJQ can be considered as a sequent calculus version of a canonical CBV λ-calculus (our modified Moggi calculus λ_{C}) in the same way that LJT has been shown [3] to be a sequent calculus version of the CBN λ-calculus.

Structure of the paper

Section 2 is a preliminary section that presents the main tools that are used to relate λ_{Q} to λ_{C}. Section 3.1 presents Plotkin's CBV λ_{V}-calculus [20] and the CPS-translations from λ_{V}. Section 3.2 identifies the target calculi $\lambda^{\mathcal{R}}_{\mathsf{CPS}}$ and $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$, and proves the confluence of $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$. Section 3.3 presents Moggi's calculus λ_{C} [18] extending λ_{V}, introducing the aforementioned variant, and in section 3.4 we prove that the Fischer translation forms a reflection in (our slightly modified version of) λ_{C} of $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$. In section 4 we present (at last) the calculus λ_{Q}, including its reduction system. Section 5 investigates the CBV semantics of λ_{Q}, via an adaptation of the Fischer CPS-translation. Section 6 combines the results of the previous sections to express the connection between λ_{Q} and λ_{C}. Section 7 concludes with directions for further work.

2 Preliminary notions

In this section we recall some concepts (mainly from [16]) about *reduction relations*, especially notions used to relate one calculus to another.

Definition 1 (Simulation) Let $\mathcal{R}$ be a total function from a set $\mathcal{A}$ to a set $\mathcal{B}$, respectively equipped with the reduction relations $\rightarrow_{\mathcal{A}}$ and $\rightarrow_{\mathcal{B}}$.

$\rightarrow_{\mathcal{B}}$ *weakly simulates* $\rightarrow_{\mathcal{A}}$ *through* $\mathcal{R}$ if for all $M, M' \in \mathcal{A}$, $M \rightarrow_{\mathcal{A}} M'$ implies $\mathcal{R}(M) \rightarrow^*_{\mathcal{B}} \mathcal{R}(M')$.

The notion of *strong simulation* is obtained by replacing $\mathcal{R}(M) \rightarrow^*_{\mathcal{B}} \mathcal{R}(M')$ by $\mathcal{R}(M) \rightarrow^+_{\mathcal{B}} \mathcal{R}(M')$ in the above definition, but in this paper we only deal with weak simulation and thus abbreviate it as "simulation".

We now define some more elaborate notions based on simulation, such as equational correspondence [24], Galois connection and reflection [16]. A useful account of these notions, with intuitive explanations, can be found for instance in [25]. In the following definition and in the rest of the paper, we write $f \cdot g$ for the functional composition that maps an element M to $g(f(M))$.

Definition 2 (Galois connection, reflection & related notions) Let $\mathcal{A}$ and $\mathcal{B}$ be sets respectively equipped with the reduction relations $\rightarrow_{\mathcal{A}}$ and $\rightarrow_{\mathcal{B}}$. Consider two mappings $f : \mathcal{A} \longrightarrow \mathcal{B}$ and $g : \mathcal{B} \longrightarrow \mathcal{A}$.

- f and g form an *equational correspondence* between $\mathcal{A}$ and $\mathcal{B}$ if the following conditions hold:
  - If $M \rightarrow_{\mathcal{A}} N$ then $f(M) \leftrightarrow^*_{\mathcal{A}} f(N)$
  - If $M \rightarrow_{\mathcal{B}} N$ then $g(M) \leftrightarrow^*_{\mathcal{A}} g(N)$

- $f \cdot g \subseteq \leftrightarrow_{\mathcal{A}}$
- $g \cdot f \subseteq \leftrightarrow_{\mathcal{B}}$

- f and g form a *Galois connection* from $\mathcal{A}$ to $\mathcal{B}$ if the following conditions hold:
  - $\rightarrow_{\mathcal{B}}$ simulates $\rightarrow_{\mathcal{A}}$ through f
  - $\rightarrow_{\mathcal{A}}$ simulates $\rightarrow_{\mathcal{B}}$ through g
  - $f \cdot g \subseteq \rightarrow_{\mathcal{A}}^*$
  - $g \cdot f \subseteq \leftarrow_{\mathcal{B}}^*$
- f and g form a *pre-Galois connection* from $\mathcal{A}$ to $\mathcal{B}$ if in the four conditions above we remove the last one.
- f and g form a *reflection* in $\mathcal{A}$ of $\mathcal{B}$ if the following conditions hold:
  - $\rightarrow_{\mathcal{B}}$ simulates $\rightarrow_{\mathcal{A}}$ through f
  - $\rightarrow_{\mathcal{A}}$ simulates $\rightarrow_{\mathcal{B}}$ through g
  - $f \cdot g \subseteq \rightarrow_{\mathcal{A}}^*$
  - $g \cdot f = \mathsf{Id}_{\mathcal{B}}$

Note that saying that f and g form an equational correspondence between $\mathcal{A}$ and $\mathcal{B}$ only means that f and g extend to a bijection between $\leftrightarrow_{\mathcal{A}}$-equivalence classes of $\mathcal{A}$ and $\leftrightarrow_{\mathcal{B}}$-equivalence classes of $\mathcal{B}$. If f and g form an equational correspondence, so do g and f; it is a symmetric relation, unlike (pre-)Galois connections and reflections.

A Galois connection provides both an equational correspondence and a pre-Galois connection. A reflection provides a Galois connection. Also note that if f and g form a reflection then g and f form a pre-Galois connection.

The notions of equational correspondence, pre-Galois connection, Galois connection, and reflection are *transitive* notions, in that, for instance, if f and g form a reflection in $\mathcal{A}$ of $\mathcal{B}$ and f' and g' form a reflection in $\mathcal{B}$ of $\mathcal{C}$ then $f \cdot f'$ and $g \cdot g'$ form a reflection in $\mathcal{A}$ of $\mathcal{C}$ (and similarly for equational correspondences, pre-Galois connection and Galois connections).

These notions based on simulation can then be used to infer, from the confluence of one calculus, the confluence of another calculus. This technique is also known as the Interpretation Method [9].

Theorem 1 (Confluence by simulation) *If f and g form a pre-Galois connection from $\mathcal{A}$ to $\mathcal{B}$ and $\rightarrow_{\mathcal{B}}$ is confluent, then $\rightarrow_{\mathcal{A}}$ is confluent.*

Proof: The proof is graphically represented in Fig. 1. □

3 Call-by-value λ-calculus

In this section we present some new developments in the theory of CBV λ-calculus.

3.1 λ_{V} & CPS-translations

We present the λ_{V}-calculus of [20]:

Definition 3 (λ_{V}) The syntax of λ_{V} is the same as that of λ-calculus, although it is elegant to present it by letting values form a syntactic category of their own:

$$\begin{array}{ll} V, W, \ldots & ::= x \mid \lambda x.M \\ M, N, \ldots & ::= V \mid M\ N \end{array}$$

$V, W, \ldots$ stand for *values*, i.e. variables or abstractions.

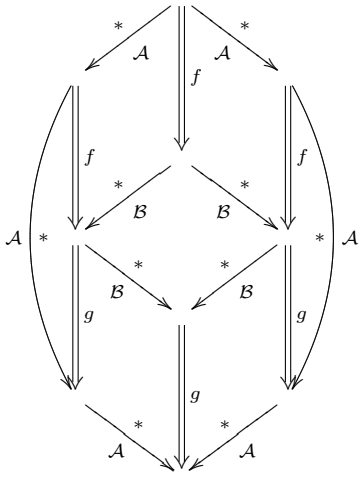

Figure 1: Confluence by simulation

We obtain the reduction rules of λ_{V} by restricting β-reduction to the cases where the argument is a value and restricting η-reduction to the cases where the function is a value:

$$\begin{array}{lllll} \beta_{\mathsf{V}} & (\lambda x.M)\ V & \longrightarrow & M\{x = V\} & \\ \eta_{\mathsf{V}} & \lambda x.V\ x & \longrightarrow & V & \text{if } x \notin \mathsf{FV}(V) \end{array}$$

In presence of β_{V}, the following rule has the same effect as η_{V}:

$$\eta'_{\mathsf{V}} \quad \lambda x.y\,x \longrightarrow y \qquad \text{if } x \neq y$$

Definition 4 ($\lambda_{\mathsf{EvalArg}}$) $\lambda_{\mathsf{EvalArg}}$ is the fragment of λ-calculus where all arguments are values, hence given by the following syntax:

$$\begin{array}{ll} V, W, \ldots & ::= x \mid \lambda x.M \\ M, N, \ldots & ::= V \mid M\ V \end{array}$$

Remark 2 (Strategy indifference) In the $\lambda_{\mathsf{EvalArg}}$ fragment, $\beta_{\mathsf{V}} = \beta$, so applying CBN- (general β) or CBV- (β_{V}) reduction is the same.

Note that the situation is different for η-conversion in $\lambda_{\mathsf{EvalArg}}$, since $\eta_{\mathsf{V}} \neq \eta$. The fragment $\lambda_{\mathsf{EvalArg}}$ is stable under β_{V}/β-reduction and under η_{V}-reduction, but not under η-reduction.

We can encode λ_{V} into $\lambda_{\mathsf{EvalArg}}$ by using the notion of continuation and defining *Continuation Passing Style translations* (*CPS-translations*). There are in fact two variants of the CBV CPS-translation: Reynolds' [21, 23] and Fischer's [7, 8], presented in Fig. 2.

Note that the two translations only differ in the order of arguments in $x\ y\ k$ / $x\ k\ y$ and $\lambda x.\lambda k.\mathcal{R}(M)\ k$ / $\lambda k.\lambda x.\mathcal{F}(M)\ k$.

As mentioned in the introduction, both translations map β_{V}-equivalent terms to β/β_{V}-equivalent terms (soundness), but the converse fails (incompleteness) (see e.g. [20]).

A more refined analysis is given by looking at the reduction rather than just equivalence. Note that the two translations above introduce many fresh variables, but bind them, often leading to new redexes and potential redexes not corresponding to redexes in the original terms. Some of these get in the way of simulating β-reduction of the original terms. However, they can be identified as *administrative*, so that the translations above can be refined by

$\mathcal{R}(V)$	$= \lambda k.k\ \mathcal{R}_{\mathsf{V}}(V)$
$\mathcal{R}(M\ N)$	$= \lambda k.\mathcal{R}(M)\ (\lambda x.\mathcal{R}(N)\ (\lambda y.x\ y\ k))$
$\mathcal{R}_{\mathsf{V}}(x)$	$= x$
$\mathcal{R}_{\mathsf{V}}(\lambda x.M)$	$= \lambda x.\lambda k.\mathcal{R}(M)\ k$

Reynolds' translation

$\mathcal{F}(V)$	$= \lambda k.k\ \mathcal{F}_{\mathsf{V}}(V)$
$\mathcal{F}(M\ N)$	$= \lambda k.\mathcal{F}(M)\ (\lambda x.\mathcal{F}(N)\ (\lambda y.x\ k\ y))$
$\mathcal{F}_{\mathsf{V}}(x)$	$= x$
$\mathcal{F}_{\mathsf{V}}(\lambda x.M)$	$= \lambda k.\lambda x.\mathcal{F}(M)\ k$

Fischer's translation

Figure 2: CBV CPS-translations, from λ_{V} to $\lambda_{\mathsf{EvalArg}}$

reducing these redexes. The precise definition of which redexes are administrative is crucial, since this choice might or might not make the refined Fischer translation a Galois connection or a reflection (Definition 2), as we shall see in section 3.4. In Fig. 3 we give the refinement for a particular choice of administrative redexes. In this figure, K ranges over λ-terms. We shall see that, for the inductive definition to work, it is sufficient to restrict the range of K to particular terms called *continuations*.

$V :_{\mathcal{R}} K$	$= K\ V^{\mathcal{R}}$	
$M\ N :_{\mathcal{R}} K$	$= M :_{\mathcal{R}} \lambda x.(x\ N :_{\mathcal{R}} K)$	if M is not a value
$V\ N :_{\mathcal{R}} K$	$= N :_{\mathcal{R}} \lambda y.(V\ y :_{\mathcal{R}} K)$	if N is not a value
$V\ V' :_{\mathcal{R}} K$	$= V^{\mathcal{R}}\ V'^{\mathcal{R}}\ K$	
$x^{\mathcal{R}}$	$= x$	
$(\lambda x.M)^{\mathcal{R}}$	$= \lambda x.\lambda k.(M :_{\mathcal{R}} k)$	

Reynolds

$V :_{\mathcal{F}} K$	$= K\ V^{\mathcal{F}}$	
$M\ N :_{\mathcal{F}} K$	$= M :_{\mathcal{F}} \lambda x.(x\ N :_{\mathcal{F}} K)$	if M is not a value
$V\ N :_{\mathcal{F}} K$	$= N :_{\mathcal{F}} \lambda y.(V\ y :_{\mathcal{F}} K)$	if N is not a value
$V\ V' :_{\mathcal{F}} K$	$= V^{\mathcal{F}}\ K\ V'^{\mathcal{F}}$	
$x^{\mathcal{F}}$	$= x$	
$(\lambda x.M)^{\mathcal{F}}$	$= \lambda k.\lambda x.(M :_{\mathcal{F}} k)$	

Fischer

Figure 3: Refined CBV CPS-translations, from $\lambda_{\mathsf{V}}$to $\lambda_{\mathsf{EvalArg}}$

We first prove that the refined translations are indeed obtained by reduction of the original ones.

Lemma 3

1. $\mathcal{R}_{\mathsf{V}}(V) \longrightarrow_\beta^* V^{\mathcal{R}}$ *and* $\mathcal{R}(M)\ K \longrightarrow_\beta^* M :_{\mathcal{R}} K$
2. $\mathcal{F}_{\mathsf{V}}(V) \longrightarrow_\beta^* V^{\mathcal{F}}$ *and* $\mathcal{F}(M)\ K \longrightarrow_\beta^* M :_{\mathcal{F}} K$

Proof: For each point the two statements are proved by mutual induction on V and M. The interesting case is for $M = M_1\ M_2$, which we present with the Fischer translation (the case of Reynolds is very similar): $\mathcal{F}(M_1\ M_2)\ K = \mathcal{F}(M_1)\ (\lambda x.\mathcal{F}(M_2)\ (\lambda y.x\ K\ y)) \longrightarrow_\beta M_1 :_{\mathcal{F}} (\lambda x.N :_{\mathcal{F}} (\lambda y.x\ K\ y))$ by induction hypothesis.

- If neither M_1 nor M_2 are values, this is also $M_1\ M_2 :_{\mathcal{F}} K$.

- If M_1 is a value and M_2 is not, this is also
  $(\lambda x.M_2 :_{\mathcal{F}} (\lambda y.x\ K\ y))\ {M_1}^{\mathcal{F}} \longrightarrow_\beta\ M_2 :_{\mathcal{F}} (\lambda y.{M_1}^{\mathcal{F}}\ K\ y) = M_1\ M_2 :_{\mathcal{F}} K$.
- If M_1 is not a value but M_2 is, this is also
  $M_1 :_{\mathcal{F}} (\lambda x.(\lambda y.x\ K\ y)\ {M_2}^{\mathcal{F}}) \longrightarrow_\beta\ M_1 :_{\mathcal{F}} (\lambda x.x\ K\ {M_2}^{\mathcal{F}}) = M_1\ M_2 :_{\mathcal{F}} K$.
- If both M_1 and M_2 are values, this is also
  $(\lambda x.(\lambda y.x\ K\ y)\ {M_2}^{\mathcal{F}})\ {M_1}^{\mathcal{F}} \longrightarrow_\beta^*\ {M_1}^{\mathcal{F}}\ K\ {M_2}^{\mathcal{F}} = M_1\ M_2 :_{\mathcal{F}} K$.

□

Remark 4 Note that K is a sub-term of $M :_{\mathcal{R}} K$ and $M :_{\mathcal{F}} K$ with exactly one occurrence[1], so for instance if $x \in \mathsf{FV}(K) \setminus \mathsf{FV}(M)$ then x has exactly one free occurrence in $M :_{\mathcal{R}} K$ and $M :_{\mathcal{F}} K$. Hence, the variables introduced by the translations and denoted by k, which we call the *continuation variables*, are such that the set of those that are free in the scope of a binder $\lambda k.$ is exactly $\{k\}$, with exactly one occurrence (only one of them can be free at a time).

In other words, in a term (which is a α-equivalence class of syntactic terms, i.e. a set) there is always a representative that always uses the same variable k. Note that K does not need to range over all λ-terms for the definition of the refined translations to make sense, but only over constructs of the form k or $\lambda x.M$, with $x \neq k$.

In that case, note that if we call *continuation redex* any β-redex binding a continuation variable (i.e. any redex of the form $(\lambda k.M)\ N$), then the refined Reynolds translation considers all continuation redexes to be administrative and has thus reduced all of them, while the refined Fischer translation leaves a continuation redex in the construct $(\lambda x.M)\ V :_{\mathcal{F}} K = (\lambda k.\lambda x.M :_{\mathcal{F}} k)\ K\ {V}^{\mathcal{F}}$, which is thus not administrative.

This choice is different from that of [25] which, as for the Reynolds translation, considers all continuation redexes to be administrative. With that choice the authors establish negative results about the refined Fischer translation as we shall discuss in section 3.4.

We can now identify the target calculi of the refined translations, i.e. the fragments of λ-calculus reached by them, and look at their associated notions of reduction.

3.2 The CPS calculi $\lambda^{\mathcal{R}}_{\mathsf{CPS}}$ & $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$

From the refined Reynolds and Fischer translations we get the target fragments of λ-calculus described in Fig. 4.

- $M, N, \ldots$ denote (CPS-)*programs*,
- $V, W, \ldots$ denote (CPS-)*values*,
- $K, K', \ldots$ denote *continuations*.

M, N	$::=$	$K\ V \mid V\ W\ K$
V, W	$::=$	$x \mid \lambda x.\lambda k.M$
		with $k \in \mathsf{FV}(M)$
K	$::=$	$k \mid \lambda x.M$

Reynolds: $\lambda^{\mathcal{R}}_{\mathsf{CPS}}$

M, N	$::=$	$K\ V \mid V\ K\ W$
V, W	$::=$	$x \mid \lambda k.\lambda x.M$
		with $k \in \mathsf{FV}(\lambda x.M)$
K	$::=$	$k \mid \lambda x.M$

Fischer: $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$

Figure 4: Target calculi

Note that values have no free occurrence of continuation variables while programs and continuations have exactly one. Also note that x ranges over an infinite set of variables, while for every term it is always possible to find a representative (i.e. a syntactic term) that uses a paricular continuation variable k. In fact we could have a constructor with arity 0 to represent this variable

[1] In some sense the construction $\_ :_{\mathcal{F}} \_$ is *linear* in its second argument.

and a constructor with arity 1 for $\lambda k.\_$, but treating k as a variable allows the use of the implicit substitution of k.

The fragments are stable under the reductions in Fig. 5 and are sufficient to simulate β_{V} and η_{V} through the CPS-translations, as we shall see in section 3.4. We write $\lambda^{\mathcal{R}}_{\mathsf{CPS}\beta}$ (resp. $\lambda^{\mathcal{F}}_{\mathsf{CPS}\beta}$) for system $\beta_{\mathsf{V}}1, \beta_{\mathsf{V}}2$ and $\lambda^{\mathcal{R}}_{\mathsf{CPS}\beta\eta}$ (resp. $\lambda^{\mathcal{F}}_{\mathsf{CPS}\beta\eta}$) for system $\beta_{\mathsf{V}}1, \beta_{\mathsf{V}}2, \eta_{\mathsf{V}}1, \eta_{\mathsf{V}}2$ in $\lambda^{\mathcal{R}}_{\mathsf{CPS}}$ (resp. $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$).

$\beta_{\mathsf{V}}1$	$(\lambda x.M)\ V$	$\longrightarrow$	$M\{x = V\}$	
$\beta_{\mathsf{V}}2$	$(\lambda x.\lambda k.M)\ V\ K$	$\longrightarrow$	$M\{x = V\}\{k = K\}$	
$\eta_{\mathsf{V}}1$	$\lambda x.\lambda k.V\ x\ k$	$\longrightarrow$	V	if $x \notin \mathsf{FV}(V)$
$\eta_{\mathsf{V}}2$	$\lambda x.K\ x$	$\longrightarrow$	K	if $x \notin \mathsf{FV}(K)$

Reynolds

$\beta_{\mathsf{V}}1$	$(\lambda x.M)\ V$	$\longrightarrow$	$M\{x = V\}$	
$\beta_{\mathsf{V}}2$	$(\lambda k.\lambda x.M)\ K\ V$	$\longrightarrow$	$(\lambda x.M\{k = K\})\ V$	
$\eta_{\mathsf{V}}1$	$\lambda k.\lambda x.V\ k\ x$	$\longrightarrow$	V	if $x \notin \mathsf{FV}(V)$
$\eta_{\mathsf{V}}2$	$\lambda x.K\ x$	$\longrightarrow$	K	if $x \notin \mathsf{FV}(K)$

Fischer

Figure 5: Reduction rules for $\lambda^{\mathcal{R}}_{\mathsf{CPS}}$ & $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$

Note the difference between the case of Reynolds and that of Fischer in the rule $\beta_{\mathsf{V}}2$. Reynolds-$\beta_{\mathsf{V}}2$ must perform two β_{V}-reduction steps, since $(\lambda k.M\{x = V\})\ K$ is not a program of the fragment. Fischer-$\beta_{\mathsf{V}}2$ performs only one β_{V}-reduction step, $(\lambda x.M\{k = K\})\ V$ being a valid program. It could obviously reduce further to $M\{k = K\}\{x = V\}$ as for the case of Reynolds, but leaving this second step as a $\beta_{\mathsf{V}}1$-reduction has nice properties: this split of reduction into two atomic β_{V}-steps makes the refined Fischer translation (as defined here) a reflection.

A good account of the refined Reynolds translation as a reflection can be found in [25], so here we study similar properties for the refined Fischer translation, building on earlier work [24] that established results of equational correspondence. Moreover, Fischer's approach helps establish connections between CBV-λ-calculus and LJQ from section 4.

We now establish the confluence of $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$. The confluence of $\lambda^{\mathcal{F}}_{\mathsf{CPS}\beta}$ is straightforward, since every case of β-reduction in $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$ is covered by either $\beta_{\mathsf{V}}1$ or $\beta_{\mathsf{V}}2$, so it is a particular case of the confluence of β-reduction in λ-calculus.

For $\lambda^{\mathcal{F}}_{\mathsf{CPS}\beta\eta}$ we use the confluence of β,η-reduction in λ-calculus, but unfortunately the language of $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$ is not stable under β,η. Fig. 6 shows its closure $\lambda^{+}_{\mathsf{CPS}}$ under β,η. Such a closure is used for instance in [24].

M, N	$::=$	$K\ V$
V, W	$::=$	$x \mid \lambda k.K$
K	$::=$	$k \mid \lambda x.M \mid V\ K$

Figure 6: Grammar of $\lambda^{+}_{\mathsf{CPS}}$

First, note that we no longer have $\beta = \beta_{\mathsf{V}}$. Second, this grammar is indeed stable under β,η; all the cases are:

$$\begin{array}{lll}
(\lambda x.M)\ V & \longrightarrow_{\beta} & M\{x = V\} \\
(\lambda k.K)\ K' & \longrightarrow_{\beta} & K\{k = K'\} \\
\lambda k.V\ k & \longrightarrow_{\eta} & V \\
\lambda x.K\ x & \longrightarrow_{\eta} & K \quad \text{if } x \notin \mathsf{FV}(K)
\end{array}$$

We can then take β,η as the reductions of $\lambda^{+}_{\mathsf{CPS}}$, and derive from the confluence of λ-calculus that β and η are confluent in $\lambda^{+}_{\mathsf{CPS}}$.

Note that $\lambda^{+}_{\mathsf{CPS}}$ is the smallest language that includes that of $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$ and that is stable under β,η: Fig. 7 defines a mapping $\Uparrow$ from $\lambda^{+}_{\mathsf{CPS}}$ onto $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$ such that

$\Uparrow M \longrightarrow_\eta M$. Our convention for parentheses assumes that $\Uparrow$ applies to the smallest expression on its right-hand side.

$\Uparrow (k\ V)$	$= k\ \Uparrow V$
$\Uparrow ((\lambda x.M)\ V)$	$= (\lambda x.\Uparrow M)\ \Uparrow V$
$\Uparrow (W\ K\ V)$	$= \Uparrow W\ \Uparrow K\ \Uparrow V$
$\Uparrow x$	$= x$
$\Uparrow \lambda k.k$	$= \lambda k.\lambda x.k\ x$
$\Uparrow \lambda k.\lambda x.M$	$= \lambda k.\lambda x.\Uparrow M$
$\Uparrow \lambda k.V\ K$	$= \lambda k.\lambda x.\Uparrow V\ \Uparrow K\ x$
$\Uparrow k$	$= k$
$\Uparrow \lambda x.M$	$= \lambda x.\Uparrow M$
$\Uparrow (V\ K)$	$= \lambda x.\Uparrow V\ \Uparrow K\ x$

Figure 7: Projection of λ^+_{CPS} onto $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$

Remark 5 Note that $\Uparrow M \longrightarrow_\eta M$, $\Uparrow V \longrightarrow_\eta V$, $\Uparrow K \longrightarrow_\eta K$ and if M, V, K are in $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$ then $\Uparrow M = M$, $\Uparrow V = V$, $\Uparrow K = K$.

We can now prove the following:

Theorem 6 ($\Uparrow$ is a Galois connection) *The identity $\mathsf{Id}_{\lambda^{\mathcal{F}}_{\mathsf{CPS}}}$ and the mapping $\Uparrow$ form a Galois connection from $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$, equipped with $\lambda^{\mathcal{F}}_{\mathsf{CPS}\beta\eta}$, to $\lambda^{+}_{\mathsf{CPS}}$, equipped with β_{V} (and also with only $\lambda^{\mathcal{F}}_{\mathsf{CPS}\beta}$ and β).*

Proof: Given Remark 5, it suffices to check the simulations:

- For the simulation of η by $\lambda^{\mathcal{F}}_{\mathsf{CPS}\beta\eta}$ through $\Uparrow$, we check all cases:

$\Uparrow \lambda k.\lambda x.k\ x$	$= \lambda k.\lambda x.k\ x$	$=$	$\Uparrow \lambda k.k$	
$\Uparrow \lambda k.\lambda x.(\lambda y.M)\ x$	$= \lambda k.\lambda x.(\lambda y.\Uparrow M)\ x$	$\longrightarrow_{\eta\mathsf{V}2}$	$\lambda k.\lambda y.\Uparrow M$	$= \Uparrow \lambda k.\lambda y.M$
$\Uparrow \lambda k.\lambda x.V\ K\ x$	$= \lambda k.\lambda x.\Uparrow V\ \Uparrow K\ x$	$=$	$\Uparrow \lambda k.V\ K$	
$\Uparrow \lambda k.V\ k$	$= \lambda k.\lambda x.\Uparrow V\ k\ x$	$\longrightarrow_{\eta\mathsf{V}1}$	$\Uparrow V$	
$\Uparrow \lambda x.k\ x$	$= \lambda x.k\ x$	$\longrightarrow_{\eta\mathsf{V}2}$	k	$= \Uparrow k$
$\Uparrow \lambda x.(\lambda y.M)\ x$	$= \lambda x.(\lambda y.\Uparrow M)\ x$	$\longrightarrow_{\eta\mathsf{V}2}$	$\lambda y.\Uparrow M$	$= \Uparrow \lambda y.M$
$\Uparrow \lambda x.V\ K\ x$	$= \lambda x.\Uparrow V\ \Uparrow K\ x$	$=$	$\Uparrow V\ K$	

For the simulation of β by $\lambda^{\mathcal{F}}_{\mathsf{CPS}\beta\eta}$ through $\Uparrow$, we must first check:

$$\begin{array}{ll} \Uparrow M\{x =\Uparrow V\} = \Uparrow M\{x = V\} & \Uparrow M\{k =\Uparrow K\} \longrightarrow^*_{\beta_{\mathsf{V}}1} \Uparrow M\{k = K\} \\ \Uparrow W\{x =\Uparrow V\} = \Uparrow W\{x = V\} & \Uparrow W\{k =\Uparrow K\} \longrightarrow^*_{\beta_{\mathsf{V}}1} \Uparrow W\{k = K\} \\ \Uparrow K'\{x =\Uparrow V\} = \Uparrow K'\{x = V\} & \Uparrow K'\{k =\Uparrow K\} \longrightarrow^*_{\beta_{\mathsf{V}}1} \Uparrow K'\{k = K\} \end{array}$$

The left-hand side equalities and the right-hand side equalities are respectively proved by mutual induction on terms, with the following interesting case:

$$\begin{array}{rll} \Uparrow (k\ V)\{k =\Uparrow K\} & = & \Uparrow K\ \Uparrow V\{k =\Uparrow K\} \\ \text{by i.h.} & \longrightarrow^*_{\beta_{\mathsf{V}}1} & \Uparrow K\ \Uparrow V\{k = K\} \\ & \longrightarrow^*_{\beta_{\mathsf{V}}1} & \Uparrow (K\ V\{k = K\}) \\ & = & \Uparrow (k\ V)\{k = K\} \end{array}$$

The penultimate step is justified by the fact that $\Uparrow K\ \Uparrow V \longrightarrow^*_{\beta_{\mathsf{V}}1} \Uparrow (K\ V)$ (it is an equality for $K = k$ or $K = \lambda x.M$ and one $\beta_{\mathsf{V}}1$-reduction step for $K = W\ K'$).

We now checks all cases for β-reduction. The last step for the simulation of the β-reduction of a value is an equality if $K'\{k = K\} = k'$ and one $\beta_{\mathsf{V}}1$-step otherwise.

$\Uparrow ((\lambda x.M)\ V)$	$=$	$(\lambda x.\Uparrow M)\ \Uparrow V$	
	$\longrightarrow_{\beta_{\mathsf{V}}1}$	$\Uparrow M\{x =\Uparrow V\}$	
	$=$	$\Uparrow M\{x = V\}$	
$\Uparrow ((\lambda k.k)\ K\ V)$	$=$	$(\lambda k.\lambda x.k\ x)\ \Uparrow K\ \Uparrow V$	
	$\longrightarrow^*_{\beta_{\mathsf{V}}2,\beta_{\mathsf{V}}1}$	$\Uparrow K\ \Uparrow V$	
	$\longrightarrow^*_{\beta_{\mathsf{V}}1}$	$\Uparrow (K\ V)$	
$\Uparrow ((\lambda k.\lambda x.M)\ K\ V)$	$=$	$(\lambda k.\lambda x.\Uparrow M)\ \Uparrow K\ \Uparrow V$	
	$\longrightarrow_{\beta_{\mathsf{V}}2}$	$(\lambda x.\Uparrow M)\{k =\Uparrow K\}\ \Uparrow V$	
	$\longrightarrow^*_{\beta_{\mathsf{V}}1}$	$\Uparrow ((\lambda x.M\{k = K\})\ V)$	
$\Uparrow ((\lambda k.W\ K')\ K\ V)$	$=$	$(\lambda k.\lambda x.\Uparrow W\ \Uparrow K'\ x)\ \Uparrow K\ \Uparrow V$	
	$\longrightarrow^*_{\beta_{\mathsf{V}}2,\beta_{\mathsf{V}}1}$	$\Uparrow W\ \Uparrow K'\{k =\Uparrow K\}\ \Uparrow V$	
	$\longrightarrow^*_{\beta_{\mathsf{V}}1}$	$\Uparrow (W\ (K'\{k = K\})\ V)$	
$\Uparrow \lambda k'.(\lambda k.K')\ K$	$=$	$\lambda k'.\lambda x.\Uparrow (\lambda k.K')\ \Uparrow K\ x$	
	$\longrightarrow^*_{\beta_{\mathsf{V}}2,\beta_{\mathsf{V}}1}$	$\lambda k'.\lambda x.\Uparrow (K'\{k = K\})\ x$	as above
	$\longrightarrow^*_{\beta_{\mathsf{V}}1}$	$\Uparrow \lambda k'.K'\{k = K\}$	
$\Uparrow ((\lambda k.K')\ K)$	$=$	$\lambda x.\Uparrow (\lambda k.K')\ \Uparrow K\ x$	
	$\longrightarrow^*_{\beta_{\mathsf{V}}2,\beta_{\mathsf{V}}1}$	$\lambda x.\Uparrow (K'\{k = K\})\ x$	as above
	$\longrightarrow_{\eta_{\mathsf{V}}2}$	$\Uparrow K'\{k = K\}$	

- The fact that β, η simulate $\beta_{\mathsf{V}}1, \beta_{\mathsf{V}}2, \eta_{\mathsf{V}}1, \eta_{\mathsf{V}}2$ through $\mathsf{Id}_{\lambda^{\mathcal{F}}_{\mathsf{CPS}}}$ is trivial, since the latter are particular cases of the former.

□

Corollary 7 (Confluence of $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$) *$\lambda^{\mathcal{F}}_{\mathsf{CPS}\beta}$ and $\lambda^{\mathcal{F}}_{\mathsf{CPS}\beta\eta}$ are confluent.*

Proof: The first system is confluent as it is the entire β-reduction relation in $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$, the second system is confluent by Theorem 6 and Theorem 1. □

3.3 Moggi's λ_{C}-calculus

Both the refined Reynolds translation and the refined Fischer translations suggest to extend λ_{V} with a construct $\mathsf{let}\ \_ = \_\ \mathsf{in}\ \_$ with the following semantics:

$$(\mathsf{let}\ x = M\ \mathsf{in}\ N){:}_{\mathcal{R}}K = M{:}_{\mathcal{R}}\lambda x.(N{:}_{\mathcal{R}}K)$$
$$(\mathsf{let}\ x = M\ \mathsf{in}\ N){:}_{\mathcal{F}}K = M{:}_{\mathcal{F}}\lambda x.(N{:}_{\mathcal{F}}K)$$

and with the following rules:

$$\begin{array}{lll} M\ N & \longrightarrow \mathsf{let}\ x = M\ \mathsf{in}\ (x\ N) & \text{if } M \text{ is not a value} \\ V\ N & \longrightarrow \mathsf{let}\ y = N\ \mathsf{in}\ (V\ y) & \text{if } N \text{ is not a value} \end{array}$$

Indeed, for both refined translations, a redex of these rules and its reduced form are mapped to the same term.

This extension is related to Moggi's monadic λ-calculus [19], which suggests additional rules, thus forming the CBV-calculus λ_{C} [18][2] defined as follows:

Definition 5 (λ_{C}) The terms of λ_{C} are given by the following grammar:

$$\begin{array}{ll} V, W, \ldots & ::= x \mid \lambda x.M \\ M, N, P, \ldots & ::= V \mid M\ N \mid \mathsf{let}\ x = M\ \mathsf{in}\ N \end{array}$$

The reduction system of λ_{C} is given in Fig. 8.

Again, η-reduction can be added:

$$\begin{array}{llll} \eta_{\mathsf{let}} & \mathsf{let}\ x = M\ \mathsf{in}\ x & \longrightarrow M & \\ \eta_{\mathsf{V}} & \lambda x.(V\ x) & \longrightarrow V & \text{if } x \notin FV(V) \end{array}$$

[2] A detailed presentation of these ideas can also be found in [25].

β_{V}	$(\lambda x.M)\ V$	$\longrightarrow$	$M\{x = V\}$
$\mathsf{let}_{\mathsf{V}}$	$\mathsf{let}\ x = V\ \mathsf{in}\ M$	$\longrightarrow$	$M\{x = V\}$
let_1	$M\ N$	$\longrightarrow$	$\mathsf{let}\ x = M\ \mathsf{in}\ (x\ N)$ (M not a value)
let_2	$V\ N$	$\longrightarrow$	$\mathsf{let}\ y = N\ \mathsf{in}\ (V\ y)$ (N not a value)
assoc	$\mathsf{let}\ y = (\mathsf{let}\ x = M\ \mathsf{in}\ N)\ \mathsf{in}\ P$	$\longrightarrow$	$\mathsf{let}\ x = M\ \mathsf{in}\ (\mathsf{let}\ y = N\ \mathsf{in}\ P)$

Figure 8: Rules of λ_{C}

And again, in presence of β_V, rule η_{V} has the same effect as the following one:

$$\eta'_{\mathsf{V}} \quad \lambda x.(y\,x) \longrightarrow y \qquad \text{if } x \neq y$$

For various purposes described in the introduction, here we also consider a slight variation of λ_{C}, in which reduction is refined by replacing the reduction rule β_{V} with the following:

$$\mathsf{B} \quad (\lambda x.M)\ N \longrightarrow \mathsf{let}\ x = N\ \mathsf{in}\ M$$

This allows the split of the rule β_{V} into two steps: B followed by $\mathsf{let}_{\mathsf{V}}$. Note that in B we do not require N to be a value. Such a restriction will only apply when reducing $\mathsf{let}\ x = N\ \mathsf{in}\ M$ by rule $\mathsf{let}_{\mathsf{V}}$.

System $\lambda_{\mathsf{C}\beta}$ is $\mathsf{B}, \mathsf{let}_{\mathsf{V}}, \mathsf{let}_1, \mathsf{let}_2, \mathsf{assoc}$ and $\lambda_{\mathsf{C}\beta\eta}$ is $\lambda_{\mathsf{C}\beta}, \eta_{\mathsf{let}}, \eta_{\mathsf{V}}$.

In [24] it is shown that, in effect, Fischer's translation forms an equational correspondence between (Moggi's original) λ_{C} and $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$. In [25], Sabry and Wadler establish not only that Reynolds' translation form an equational correspondence between (Moggi's original) λ_{C} and $\lambda^{\mathcal{R}}_{\mathsf{CPS}}$, but the refined Reynolds translation actually forms a reflection.

3.4 The refined Fischer translation *is* a reflection

In [25], Sabry and Wadler also establish that for a particular definition of administrative redex (namely, every β-redex with a binder on the continuation variable k is administrative), the refined Fischer translation cannot be a reflection, and from λ_{C} to $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$ it cannot even be a Galois connection.

Here we show that our (different) choice of administrative redex for the Fischer translation (given in Fig. 3) makes it a reflection of $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$ in our version of λ_{C}, where the rule β_{V} is decomposed into two steps as described above. This will also bring λ_{C} closer to LJQ.

Lemma 8

1. $(M :_{\mathcal{F}} K)\{k = K'\} = M :_{\mathcal{F}} K\{k = K'\}$
2. $(M :_{\mathcal{F}} K)\{x = V^{\mathcal{F}}\} = M\{x = V\} :_{\mathcal{F}} K\{x = V^{\mathcal{F}}\}$ *and* $W^{\mathcal{F}}\{x = V^{\mathcal{F}}\} = (W\{x = V\})^{\mathcal{F}}$.
3. *If* $K \longrightarrow_{\lambda^{\mathcal{F}}_{\mathsf{CPS}\beta}} K'$ *then* $M :_{\mathcal{F}} K \longrightarrow_{\lambda^{\mathcal{F}}_{\mathsf{CPS}\beta}} M :_{\mathcal{F}} K'$
   (and similarly for $\longrightarrow_{\lambda^{\mathcal{F}}_{\mathsf{CPS}\beta\eta}}$ *).*

Proof: Straightforward induction on M for the first and third points and on M and W for the second. □

Theorem 9 (Simulation of λ_{C} in $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$) *The reduction relation $\longrightarrow_{\lambda_{\mathbf{C}\beta}}$ (resp. $\longrightarrow_{\lambda_{\mathbf{C}\beta\eta}}$) is simulated by $\longrightarrow_{\lambda^{\mathcal{F}}_{\mathbf{CPS}\beta}}$ (resp. $\longrightarrow_{\lambda^{\mathcal{F}}_{\mathbf{CPS}\beta\eta}}$) through the refined Fischer translation.*

Proof: By induction on the size of the term being reduced: We check all the root reduction cases, relying on Lemma 8:

$((\lambda x.M)\ V){:}_{\mathcal{F}}K$		$=$	$(\lambda k.\lambda x.(M{:}_{\mathcal{F}}k))\ K\ V^{\mathcal{F}}$
	(Lemma 8.1)	$\longrightarrow_{\beta_{\mathbf{V}}2}$	$(\lambda x.(M{:}_{\mathcal{F}}K))\ V^{\mathcal{F}}$
		$=$	$(\mathsf{let}\ x = V\ \mathsf{in}\ M){:}_{\mathcal{F}}K$
$((\lambda x.M)\ N){:}_{\mathcal{F}}K$		$=$	$N{:}_{\mathcal{F}}\lambda y.(\lambda k.\lambda x.(M{:}_{\mathcal{F}}k))\ K\ y$
(N not a value)	(Lemma 8)	$\longrightarrow^{*}_{\beta_{\mathbf{V}}2,\beta_{\mathbf{V}}1}$	$N{:}_{\mathcal{F}}\lambda x.(M{:}_{\mathcal{F}}K)$
		$=$	$(\mathsf{let}\ x = N\ \mathsf{in}\ M){:}_{\mathcal{F}}K$
$(\mathsf{let}\ x = V\ \mathsf{in}\ M){:}_{\mathcal{F}}K$		$=$	$(\lambda x.M{:}_{\mathcal{F}}K)\ V^{\mathcal{F}}$
		$\longrightarrow_{\beta_{\mathbf{V}}1}$	$(M{:}_{\mathcal{F}}K)\{x = V^{\mathcal{F}}\}$
	(Lemma 8.2)	$=$	$(M\{x = V\}){:}_{\mathcal{F}}K$
$(M\ N){:}_{\mathcal{F}}K$		$=$	$M{:}_{\mathcal{F}}\lambda x.(x\ N{:}_{\mathcal{F}}K)$
(M not a value)		$=$	$(\mathsf{let}\ x = M\ \mathsf{in}\ x\ N){:}_{\mathcal{F}}K$
$(V\ N){:}_{\mathcal{F}}K$		$=$	$N{:}_{\mathcal{F}}\lambda x.(V\ x{:}_{\mathcal{F}}K)$
(N not a value)		$=$	$(\mathsf{let}\ x = N\ \mathsf{in}\ V\ x){:}_{\mathcal{F}}K$
$(\mathsf{let}\ y = (\mathsf{let}\ x = M\ \mathsf{in}\ N)\ \mathsf{in}\ P){:}_{\mathcal{F}}K$		$=M{:}_{\mathcal{F}}\lambda x.(N{:}_{\mathcal{F}}\lambda y.(P{:}_{\mathcal{F}}K))$	
		$=(\mathsf{let}\ x = M\ \mathsf{in}\ (\mathsf{let}\ y = N\ \mathsf{in}\ P)){:}_{\mathcal{F}}K$	
$(\mathsf{let}\ x = M\ \mathsf{in}\ x){:}_{\mathcal{F}}K$		$=$	$M{:}_{\mathcal{F}}\lambda x.K\ x$
	(Lemma 8.3)	$\longrightarrow_{\eta_{\mathbf{V}}2}$	$M{:}_{\mathcal{F}}K$
$(\lambda x.V\ x)^{\mathcal{F}}$		$=$	$\lambda k.\lambda x.V^{\mathcal{F}}\ k\ x$
		$\longrightarrow_{\eta_{\mathbf{V}}1}$	$V^{\mathcal{F}}$

The contextual closure is straightforward as well: only the side-condition "N is not a value" can become false by reduction of N. In that case if $N \longrightarrow_{\lambda_{\mathbf{C}\beta}} V$ we have

$$\begin{array}{rll} N\ M{:}_{\mathcal{F}}K & = & N{:}_{\mathcal{F}}\lambda x.(x\ M{:}_{\mathcal{F}}K) \\ \text{by i.h.} & \longrightarrow^{*}_{\lambda^{\mathcal{F}}_{\mathbf{CPS}\beta}} & V{:}_{\mathcal{F}}\lambda x.(x\ M{:}_{\mathcal{F}}K) \\ & = & (\lambda x.(x\ M{:}_{\mathcal{F}}K))\ V^{\mathcal{F}} \\ & \longrightarrow_{\beta_{\mathbf{V}}1} & V\ M{:}_{\mathcal{F}}K \end{array}$$

as well as:

$$\begin{array}{rll} W\ N{:}_{\mathcal{F}}K & = & N{:}_{\mathcal{F}}\lambda x.(W\ x{:}_{\mathcal{F}}K) \\ \text{by i.h.} & \longrightarrow^{*}_{\lambda^{\mathcal{F}}_{\mathbf{CPS}\beta}} & V{:}_{\mathcal{F}}\lambda x.(W\ x{:}_{\mathcal{F}}K) \\ & = & (\lambda x.(W\ x{:}_{\mathcal{F}}K))\ V^{\mathcal{F}} \\ & \longrightarrow_{\beta_{\mathbf{V}}1} & W\ V{:}_{\mathcal{F}}K \end{array}$$

and also if M is not a value:

$$\begin{array}{rll} M\ N{:}_{\mathcal{F}}K & = & M{:}_{\mathcal{F}}\lambda x.(x\ N{:}_{\mathcal{F}}K) \\ \text{by i.h.} & \longrightarrow^{*}_{\lambda^{\mathcal{F}}_{\mathbf{CPS}\beta}} & M{:}_{\mathcal{F}}\lambda x.(x\ V{:}_{\mathcal{F}}K) \\ & = & M\ V{:}_{\mathcal{F}}K \end{array}$$

and similarly with $\longrightarrow_{\lambda^{\mathcal{F}}_{\mathbf{CPS}\beta\eta}}$ instead of $\longrightarrow_{\lambda^{\mathcal{F}}_{\mathbf{CPS}\beta}}$. □

Definition 6 (The Fischer reverse translation)
We define a translation from $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$ to λ_{C}:

$(k\ V)^{\mathcal{F}\mathsf{back}}$	$= V^{\mathcal{F}\mathsf{back}}$
$((\lambda x.M)\ V)^{\mathcal{F}\mathsf{back}}$	$= \mathsf{let}\ x = V^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ M^{\mathcal{F}\mathsf{back}}$
$(W\ k\ V)^{\mathcal{F}\mathsf{back}}$	$= W^{\mathcal{F}\mathsf{back}}\ V^{\mathcal{F}\mathsf{back}}$
$(W\ (\lambda x.M)\ V)^{\mathcal{F}\mathsf{back}}$	$= \mathsf{let}\ x = W^{\mathcal{F}\mathsf{back}}\ V^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ M^{\mathcal{F}\mathsf{back}}$
$x^{\mathcal{F}\mathsf{back}}$	$= x$
$(\lambda k.\lambda x.M)^{\mathcal{F}\mathsf{back}}$	$= \lambda x.M^{\mathcal{F}\mathsf{back}}$

Lemma 10

1. $(W\{x = V\})^{\mathcal{F}\mathsf{back}} = W^{\mathcal{F}\mathsf{back}}\{x = V^{\mathcal{F}\mathsf{back}}\}$ *and*
   $(M\{x = V\})^{\mathcal{F}\mathsf{back}} = M^{\mathcal{F}\mathsf{back}}\{x = V^{\mathcal{F}\mathsf{back}}\}$.
2. $\mathsf{let}\ x = M^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ N^{\mathcal{F}\mathsf{back}} \longrightarrow^*_{\lambda_{\mathsf{C}\beta}} (M\{k = \lambda x.N\})^{\mathcal{F}\mathsf{back}}$ *(if* $k \in \mathsf{FV}(M)$*).*

Proof: The first point is straightforward by induction on W, M. The second is proved by induction on M:

$$
\begin{array}{lll}
\mathsf{let}\ x = (k\ V)^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ N^{\mathcal{F}\mathsf{back}} & & \\
\qquad = & \mathsf{let}\ x = V^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ N^{\mathcal{F}\mathsf{back}} & \\
\qquad = & ((k\ V)\{k = \lambda x.N\})^{\mathcal{F}\mathsf{back}} & \\
\mathsf{let}\ x = ((\lambda y.P)\ V)^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ N^{\mathcal{F}\mathsf{back}} & & \\
\qquad = & \mathsf{let}\ x = (\mathsf{let}\ y = V^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ P^{\mathcal{F}\mathsf{back}})\ \mathsf{in}\ N^{\mathcal{F}\mathsf{back}} & \\
\qquad \longrightarrow_{\mathsf{assoc}} & \mathsf{let}\ y = V^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ \mathsf{let}\ x = P^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ N^{\mathcal{F}\mathsf{back}} & \\
\text{by i.h.} \quad \longrightarrow^*_{\lambda_{\mathsf{C}\beta}} & \mathsf{let}\ y = V^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ (P\{k = \lambda x.N\})^{\mathcal{F}\mathsf{back}} & \\
\qquad = & ((\lambda y.P\{k = \lambda x.N\})\ V)^{\mathcal{F}\mathsf{back}} & \\
\mathsf{let}\ x = (W\ k\ V)^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ N^{\mathcal{F}\mathsf{back}} & & \\
\qquad = & \mathsf{let}\ x = W^{\mathcal{F}\mathsf{back}}\ V^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ N^{\mathcal{F}\mathsf{back}} & \\
\qquad = & (W\ (\lambda x.N)\ V)^{\mathcal{F}\mathsf{back}} & \\
\qquad = & ((W\ k\ V)\{k = \lambda x.N\})^{\mathcal{F}\mathsf{back}} & \\
\mathsf{let}\ x = (W\ (\lambda y.P)\ V)^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ N^{\mathcal{F}\mathsf{back}} & & \\
\qquad = & \mathsf{let}\ x = (\mathsf{let}\ y = W^{\mathcal{F}\mathsf{back}}\ V^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ P^{\mathcal{F}\mathsf{back}})\ \mathsf{in}\ N^{\mathcal{F}\mathsf{back}} & \\
\qquad \longrightarrow_{\mathsf{assoc}} & \mathsf{let}\ y = W^{\mathcal{F}\mathsf{back}}\ V^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ \mathsf{let}\ x = P^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ N^{\mathcal{F}\mathsf{back}} & \\
\text{by i.h.} \quad \longrightarrow^*_{\lambda_{\mathsf{C}\beta}} & \mathsf{let}\ y = W^{\mathcal{F}\mathsf{back}}\ V^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ (P^{\mathcal{F}\mathsf{back}}\{k = \lambda x.N\}) & \\
\qquad = & (W\ (\lambda y.P^{\mathcal{F}\mathsf{back}}\{k = \lambda x.N\})\ V)^{\mathcal{F}\mathsf{back}} & \\
\qquad = & ((W\ (\lambda y.P)\ V)\{k = \lambda x.N\})^{\mathcal{F}\mathsf{back}} &
\end{array}
$$

□

Theorem 11 (Simulation of $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$ in λ_{C}) *The reduction relation* $\longrightarrow_{\lambda^{\mathcal{F}}_{\mathsf{CPS}\beta}}$ *(resp.* $\longrightarrow_{\lambda^{\mathcal{F}}_{\mathsf{CPS}\beta\eta}}$*) is simulated by* $\longrightarrow_{\lambda_{\mathsf{C}\beta}}$ *(resp.* $\longrightarrow_{\lambda_{\mathsf{C}\beta\eta}}$*) through* $\mathcal{F}\mathsf{back}$.

Proof: By induction on the size of the term being reduced: We check all root reduction cases, relying on Lemma 10:

$$
\begin{array}{lll}
((\lambda x.M)\ V)^{\mathcal{F}\mathsf{back}} & = & \mathsf{let}\ x = V^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ M^{\mathcal{F}\mathsf{back}} \\
& \longrightarrow_{\mathsf{let_V}} & M^{\mathcal{F}\mathsf{back}}\{x = V^{\mathcal{F}\mathsf{back}}\} \\
\text{(Lemma 10)} & = & (M\{x = V\})^{\mathcal{F}\mathsf{back}} \\
((\lambda k.\lambda x.M)\ k'\ V)^{\mathcal{F}\mathsf{back}} & = & (\lambda x.M^{\mathcal{F}\mathsf{back}})\ V^{\mathcal{F}\mathsf{back}} \\
& \longrightarrow_{\mathsf{B}} & \mathsf{let}\ x = V^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ M^{\mathcal{F}\mathsf{back}} \\
& = & ((\lambda x.M\{k = k'\})\ V)^{\mathcal{F}\mathsf{back}} \\
((\lambda k.\lambda x.M)\ (\lambda y.N)\ V)^{\mathcal{F}\mathsf{back}} & & \\
& = & \mathsf{let}\ y = (\lambda x.M^{\mathcal{F}\mathsf{back}})\ V^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ N^{\mathcal{F}\mathsf{back}} \\
& \longrightarrow_{\mathsf{B}} & \mathsf{let}\ y = (\mathsf{let}\ x = V^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ M^{\mathcal{F}\mathsf{back}})\ \mathsf{in}\ N^{\mathcal{F}\mathsf{back}} \\
& = & \mathsf{let}\ y = ((\lambda x.M)\ V)^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ N^{\mathcal{F}\mathsf{back}} \\
\text{(Lemma 10)} & \longrightarrow_{\lambda_{\mathsf{C}\beta}} & (((\lambda x.M)\ V)\{k = \lambda y.N^{\mathcal{F}\mathsf{back}}\})^{\mathcal{F}\mathsf{back}} \\
\hline
(\lambda k.\lambda x.V\ k\ x)^{\mathcal{F}\mathsf{back}} & = & \lambda x.V^{\mathcal{F}\mathsf{back}}\ x \\
& \longrightarrow_{\eta_{\mathsf{V}}} & V^{\mathcal{F}\mathsf{back}} \\
((\lambda x.K\ x)\ V)^{\mathcal{F}\mathsf{back}} & \longrightarrow_{\lambda_{\mathsf{C}\beta}} & (K\ V)^{\mathcal{F}\mathsf{back}} \qquad \text{by simulation of } \beta_{\mathsf{V}}1 \\
(W\ (\lambda x.k\ x)\ V)^{\mathcal{F}\mathsf{back}} & = & \mathsf{let}\ x = W^{\mathcal{F}\mathsf{back}}\ V^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ x \\
& \longrightarrow_{\eta_{\mathsf{let}}} & W^{\mathcal{F}\mathsf{back}}\ V^{\mathcal{F}\mathsf{back}} \\
& = & (W\ k\ V)^{\mathcal{F}\mathsf{back}} \\
(W\ (\lambda x.(\lambda y.M)\ x)\ V)^{\mathcal{F}\mathsf{back}} & & \\
& = & \mathsf{let}\ x = W^{\mathcal{F}\mathsf{back}}\ V^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ \mathsf{let}\ y = x\ \mathsf{in}\ M^{\mathcal{F}\mathsf{back}} \\
& \longrightarrow_{\mathsf{let_V}} & \mathsf{let}\ y = W^{\mathcal{F}\mathsf{back}}\ V^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ M^{\mathcal{F}\mathsf{back}} \\
& = & (W\ (\lambda y.M)\ V)^{\mathcal{F}\mathsf{back}}
\end{array}
$$

□

Lemma 12 *$V \longrightarrow^*_{\lambda_{\mathsf{C}\beta}} V^{\mathcal{F}\mathcal{F}\mathsf{back}}$ and $M \longrightarrow^*_{\lambda_{\mathsf{C}\beta}} (M :_{\mathcal{F}} k)^{\mathcal{F}\mathsf{back}}$.*

Proof: By induction on the size of V, M. The cases for V ($V = x$ and $V = \lambda x.M$) are straightforward, as is the case of $M = V$. For $M = (\mathsf{let}\ x = N'\ \mathsf{in}\ N)$ we have:

$$\begin{array}{lll} M & = & (\mathsf{let}\ x = N'\ \mathsf{in}\ N) \\ & \longrightarrow_{\lambda_{\mathsf{C}\beta}} & (\mathsf{let}\ x = (N' :_{\mathcal{F}} k)^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ (N :_{\mathcal{F}} k)^{\mathcal{F}\mathsf{back}}) \quad \text{by i.h.} \\ & \longrightarrow_{\lambda_{\mathsf{C}\beta}} & ((N' :_{\mathcal{F}} k)\{k = \lambda x.N :_{\mathcal{F}} k\})^{\mathcal{F}\mathsf{back}} \quad \text{(Lemma 10)} \\ & = & (N' :_{\mathcal{F}} \lambda x.(N :_{\mathcal{F}} k))^{\mathcal{F}\mathsf{back}} \quad \text{(Lemma 8)} \\ & = & ((\mathsf{let}\ x = N'\ \mathsf{in}\ N) :_{\mathcal{F}} k)^{\mathcal{F}\mathsf{back}} \end{array}$$

The cases for $M = M_1\ M_2$ are as follows:

$$\begin{array}{lll} M_1\ M_2 & \longrightarrow_{\lambda_{\mathsf{C}\beta}} & \mathsf{let}\ y = M_1\ \mathsf{in}\ y\ M_2 \\ (M_1 \text{ not a value}) & \longrightarrow^*_{\lambda_{\mathsf{C}\beta}} & \mathsf{let}\ y = (M_1 :_{\mathcal{F}} k)^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ (y\ M_2 :_{\mathcal{F}} k)^{\mathcal{F}\mathsf{back}} \quad \text{by i.h.} \\ & \longrightarrow^*_{\lambda_{\mathsf{C}\beta}} & (M_1 :_{\mathcal{F}} \lambda y.(y\ M_2 :_{\mathcal{F}} k))^{\mathcal{F}\mathsf{back}} \quad \text{(Lemma 10)} \\ & = & ((M_1\ M_2) :_{\mathcal{F}} k)^{\mathcal{F}\mathsf{back}} \\ V\ M_2 & \longrightarrow_{\lambda_{\mathsf{C}\beta}} & \mathsf{let}\ y = M_2\ \mathsf{in}\ V\ y \\ (M_2 \text{ not a value}) & \longrightarrow^*_{\lambda_{\mathsf{C}\beta}} & \mathsf{let}\ y = (M_2 :_{\mathcal{F}} k)^{\mathcal{F}\mathsf{back}}\ \mathsf{in}\ (V\ y :_{\mathcal{F}} k)^{\mathcal{F}\mathsf{back}} \quad \text{by i.h.} \\ & \longrightarrow^*_{\lambda_{\mathsf{C}\beta}} & (M_2 :_{\mathcal{F}} \lambda y.(V\ y :_{\mathcal{F}} k))^{\mathcal{F}\mathsf{back}} \quad \text{(Lemma 10)} \\ & = & ((V\ M_2) :_{\mathcal{F}} k)^{\mathcal{F}\mathsf{back}} \\ V\ W & \longrightarrow^*_{\lambda_{\mathsf{C}\beta}} & V^{\mathcal{F}\mathcal{F}\mathsf{back}}\ W^{\mathcal{F}\mathcal{F}\mathsf{back}} \quad \text{by i.h.} \\ & = & (V^{\mathcal{F}}\ k\ W^{\mathcal{F}})^{\mathcal{F}\mathsf{back}} \\ & = & (V\ W :_{\mathcal{F}} k)^{\mathcal{F}\mathsf{back}} \end{array}$$

□

Lemma 13 *$V = V^{\mathcal{F}\mathsf{back}^{\mathcal{F}}}$ and $M = M^{\mathcal{F}\mathsf{back}} :_{\mathcal{F}} k$.*

Proof: Straightforward induction on V, M. □

Now we can prove the following:

Theorem 14 (The refined Fischer translation is a reflection)
The refined Fischer translation and $\mathcal{F}\mathsf{back}$ form a reflection in λ_{C} of $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$.

Proof: This theorem is just the conjunction of Theorem 9, Theorem 11, Lemma 12 and Lemma 13. □

Corollary 15 (Confluence of λ_{C}) *$\lambda_{\mathsf{C}\beta}$ and $\lambda_{\mathsf{C}\beta\eta}$ are confluent.*

Proof: By Theorem 14 and Theorem 1. □

4 A Term Calculus for LJQ: λ_{Q}

LJQ was described in [4]; here we show it only as the typing system of a term calculus λ_{Q} (called $\lambda\mathsf{LJQ}$ in [4]). We then establish a connection between λ_{Q} and the (modified) call-by-value λ-calculus λ_{C} of Moggi discussed above.

The terms of this calculus are given by the following grammar:

$$\begin{array}{lll} V, V' & ::= & x \mid \lambda x.M \mid C_1(V, x.V') \\ M, N, P & ::= & [V] \mid x(V, y.N) \mid C_2(V, x.N) \mid C_3(M, x.N) \end{array}$$

The terms $C_i(-, -.-)$ are *explicit substitutions*, typed by *Cut* rules, and are to be distinguished from the meta-notation $M\{x = N\}$ standing for "M with

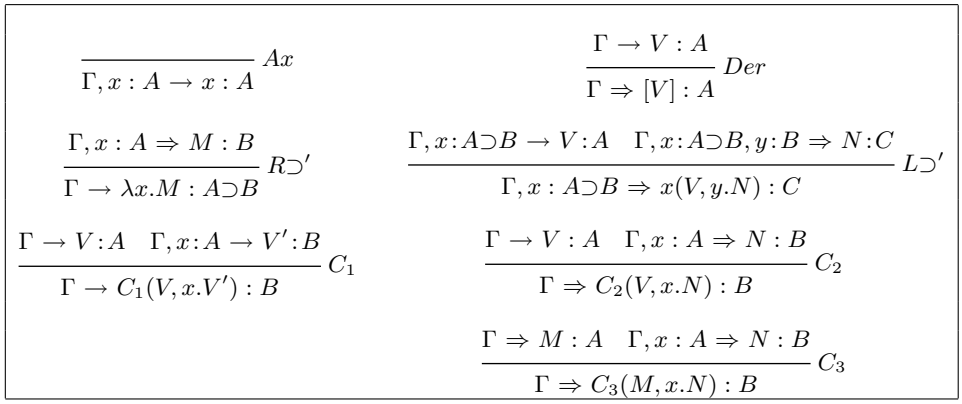

Figure 9: LJQ with terms

x replaced by N". Binding occurrences of variables are those immediately followed by ".". Contexts Γ are finite mappings from variables to formulae/types, represented in the usual way.

Note that the grammar has two syntactic categories: *values* and *terms*. They provide proof-terms for *focussed* or *unfocussed* sequents of LJQ, respectively. This can be seen in the typing rules shown in Fig. 9, which connect the above grammar to the focused sequent calculus LJQ of [4]. A naive explanation of the different meanings of the two types of arrow was given in the introduction.

This typing system is as given in [4]. Compared to other presentations, such as in [10], it omits non-implicational connectives, uses proof terms, uses two kinds of sequent, has the formula $A \supset B$ in the second premiss of $L\supset'$ and uses context-sharing cut rules.

$C_3([\lambda x.M], y.y(V, z.P))$	$\longrightarrow$	$C_3(C_3([V], x.M), z.P)$ if $y \notin \mathsf{FV}(V) \cup \mathsf{FV}(P)$
$C_3([x], y.N)$	$\longrightarrow$	$N\{y = x\}$
$C_3(M, y.[y])$	$\longrightarrow$	M
$C_3(z(V, y.P), x.N)$	$\longrightarrow$	$z(V, y.C_3(P, x.N))$
$C_3(C_3([V]', y.y(V, z.P)), x.N)$	$\longrightarrow$	$C_3([V'], y.y(V, z.C_3(P, x.N)))$ if $y \notin \mathsf{FV}(V) \cup \mathsf{FV}(P)$
$C_3(C_3(M, y.P), x.N)$	$\longrightarrow$	$C_3(M, y.C_3(P, x.N))$ if the redex is not one of the previous rule
$C_3([\lambda y.M], x.N)$	$\longrightarrow$	$C_2(\lambda y.M, x.N)$ if N is not an x-covalue (see below)
$C_1(V, x.x)$	$\longrightarrow$	V
$C_1(V, x.y)$	$\longrightarrow$	y
$C_1(V, x.\lambda y.M)$	$\longrightarrow$	$\lambda y.C_2(V, x.M)$
$C_2(V, x.[V'])$	$\longrightarrow$	$[C_1(V, x.V')]$
$C_2(V, x.x(V', z.P))$	$\longrightarrow$	$C_3([V], x.x(C_1(V, x.V'), z.C_2(V, x.P)))$
$C_2(V, x.x'(V', z.P))$	$\longrightarrow$	$x'(C_1(V, x.V'), z.C_2(V, x.P))$
$C_2(V, x.(C_3(M, y.P)))$	$\longrightarrow$	$C_3(C_2(V, x.M), y.C_2(V, x.P))$

N is an *x-covalue* iff $N = [x]$ or N is of the form $x(V, z.P)$ with $x \notin \mathsf{FV}(V) \cup \mathsf{FV}(P)$

Figure 10: LJQ-reductions

The reduction rules for the calculus are shown in Fig. 10. This reduction system has the following properties:

1. It reduces any term that contains an explicit substitution;
2. It satisfies the *Subject Reduction* property;
3. It is confluent;
4. It is *Strongly Normalising*;
5. A fortiori, it is *Weakly Normalising*.

As a corollary of 1, 2 and 5, we have the admissibility of *Cut*. It is interesting to see in the proof of *Subject Reduction* how these reductions transform the proof derivations and to compare them to those used in inductive proofs of Cut-admissibility such as that of [4].

The reduction system here is more subtle, because we are interested not only in its weak normalisation but also in its strong normalisation and its connection with call-by-value λ-calculus. Indeed, the first reduction rule, breaking a cut on an implication into cuts on its direct sub-formulae, is done with C_3, although the use of C_2 would seem simpler. The reason is that we use C_3 to encode each β-redex of λ-calculus and C_2 to simulate the evaluation of its substitutions. Just as in λ-calculus, where substitutions can be pushed through β-redexes, so may C_2 be pushed through C_3, by use of the penultimate rule of Fig. 10 (which is not needed if the only concern is cut-admissibility).

In the rest of this paper we investigate the relation between λ_{Q} and CBV λ-calculus by means of simulation techniques introduced in Sect. 2. In particular we investigate the semantics of λ_{Q} using *continuation-passing style* translations, from which we infer the confluence of λ_{Q}. All the results about λ_{Q} presented hereafter can be considered independently from typing.

We leave the normalisation results about typed λ_{Q} for another paper, since the proofs require a slightly different framework.

5 The CPS-semantics of λ_{Q}

5.1 From λ_{Q} to λ_{CPS}

We can adapt Fischer's translation to λ_{Q} so that reductions in λ_{Q} can be simulated. The (refined) Fischer CPS-translation of the terms of λ_{Q} is presented in Fig. 11.

$[V]\colon K$	$=$	$K\ V^\dagger$	
$(x(V, y.M))\colon K$	$=$	$x\ (\lambda y.(M\colon K))\ V^\dagger$	
$C_3(W, x.x(V, y.M))\colon K$	$=$	$W^\dagger\ (\lambda y.(M\colon K))\ V^\dagger$	if $x \notin \mathsf{FV}(V) \cup \mathsf{FV}(M)$
$(C_3(N, x.M))\colon K$	$=$	$N\colon \lambda x.(M\colon K)$	otherwise
$(C_2(V, x.M))\colon K$	$=$	$(M\colon K)\{x = V^\dagger\}$	
$x^\dagger$	$=$	x	
$(\lambda x.M)^\dagger$	$=$	$\lambda k.\lambda x.(M\colon k)$	
$(C_1(V, x.W))^\dagger$	$=$	$W^\dagger\{x = V^\dagger\}$	

Figure 11: The (refined) Fischer translation from λ_{Q}

Now we prove the simulation of λ_{Q} by $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$, and for that we need the following remark and lemma.

Remark 16 $\mathsf{FV}(M\colon K) \subseteq \mathsf{FV}(M) \cup \mathsf{FV}(K)$ and $\mathsf{FV}(V^\dagger) \subseteq \mathsf{FV}(V)$.

Lemma 17

1. $(M\colon K)\{k = K'\} = M\colon K\{k = K'\}$.
2. $M\colon \lambda x.(P\colon K) \longrightarrow^*_{\beta_{\mathbf{V}1}} C_3(M, x.P)\colon K$ *if* $x \notin \mathsf{FV}(K)$.

3. $(V\{x=y\})^\dagger = V^\dagger\{x=y\}$, *and*
   $M\{x=y\}\colon K = (M\colon K)\{x=y\}$, *provided* $x \notin \mathsf{FV}(K)$.
4. *If* $x \notin \mathsf{FV}(K)$, *then* $M\colon \lambda x.K\ x \longrightarrow_{\beta_{\mathsf{V}}1} M\colon K$.
5. $(M\colon K)\{x=V^\dagger\} = (M\colon K\{x=V^\dagger\})\{x=V^\dagger\}$

Proof:

1. By induction on M.
2. The interesting case is the following:

$$\begin{array}{lll} W\colon \lambda x.(x(V,y.M)\colon K) & = & (\lambda x.x\ (\lambda y.(M\colon K))\ V^\dagger)\ W^\dagger \\ & \longrightarrow_{\beta_{\mathsf{V}}1} & W^\dagger\ (\lambda y.(M\colon K))\ V^\dagger \\ & = & (C_3(W, x.x(V,y.M)))\colon K \end{array}$$

   when $x \notin \mathsf{FV}(V) \cup \mathsf{FV}(M)$.
3. By structural induction on V, M.
4. By induction on M. The translation propagates the continuation $\lambda x.K\ x$ into the sub-terms of M until it reaches a value, for which $[V]\colon \lambda x.K\ x = (\lambda x.K\ x)\ V^\dagger \longrightarrow_{\beta_{\mathsf{V}}1} K\ V = [V]\colon K$.
5. By induction on M. The term $M\colon K$ only depends on K in that K is a sub-term of $M\colon K$, affected by the substitution.

□

Theorem 18 (Simulation of λ_{Q})

1. *If* $M \longrightarrow_{\lambda_{\mathsf{Q}}} M'$ *then for all* K *we have* $M\colon K \longrightarrow^*_{\lambda^{\mathcal{F}}_{\mathsf{CPS}\beta}} M'\colon K$.
2. *If* $W \longrightarrow_{\lambda_{\mathsf{Q}}} W'$ *then we have* $W^\dagger \longrightarrow^*_{\lambda^{\mathcal{F}}_{\mathsf{CPS}\beta}} W'^\dagger$.

Proof: By simultaneous induction on the derivation of the reduction step, using Lemma 17. □

5.2 A restriction on $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$: $\lambda^{f}_{\mathsf{CPS}}$

The (refined) Fischer translation of λ_{Q} is not surjective on the terms of λ_{CPS}, indeed we only need the terms of Fig. 12, which we call $\lambda^{f}_{\mathsf{CPS}}$.

M, N	$::= K\ V \mid V\ (\lambda x.M)\ W$	
V, W	$::= x \mid \lambda k.\lambda x.M$	$k \in \mathsf{FV}(M)$
K	$::= k \mid \lambda x.M$	

Figure 12: $\lambda^{f}_{\mathsf{CPS}}$

Note that $\lambda^{f}_{\mathsf{CPS}}$ is stable under $\beta_{\mathsf{V}}1, \beta_{\mathsf{V}}2$, but not under $\eta_{\mathsf{V}}1$ and $\eta_{\mathsf{V}}2$. However we can equip $\lambda^{f}_{\mathsf{CPS}}$ with the reduction system of Fig. 13. Note that $\beta_{\mathsf{V}}1$ is the same as for $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$ and $\beta_{\mathsf{V}}3$ is merely rule $\beta_{\mathsf{V}}2$ with the assumption that the redex is in $\lambda^{f}_{\mathsf{CPS}}$, while rule $\eta_{\mathsf{V}}3$ combines $\eta_{\mathsf{V}}2$ and $\eta_{\mathsf{V}}1$ in one step. We write $\lambda^{f}_{\mathsf{CPS}\beta}$ for system $\beta_{\mathsf{V}}1, \beta_{\mathsf{V}}2$ and $\lambda^{f}_{\mathsf{CPS}\beta\eta}$ for system $\beta_{\mathsf{V}}1, \beta_{\mathsf{V}}2, \eta_{\mathsf{V}}1, \eta_{\mathsf{V}}2$ in $\lambda^{f}_{\mathsf{CPS}}$.

$(\lambda x.M)\ V$	$\longrightarrow_{\beta_{\mathsf{V}}1}$	$M\{x=V\}$	
$(\lambda k.\lambda x.M)\ (\lambda y.N)\ V$	$\longrightarrow_{\beta_{\mathsf{V}}3}$	$(\lambda x.M\{k=\lambda y.N\})\ V$	
$\lambda k.\lambda x.V\ (\lambda z.k\ z)\ x$	$\longrightarrow_{\eta_{\mathsf{V}}3}$	V	if $x \notin \mathsf{FV}(V)$

Figure 13: Reduction rules of $\lambda^{f}_{\mathsf{CPS}}$

We can now project $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$ onto $\lambda^{f}_{\mathsf{CPS}}$, as shown in Fig. 14.

$\uparrow (K\ V)$	$= \uparrow K\ \uparrow V$
$\uparrow (W\ k\ V)$	$= \uparrow W\ (\lambda x.k\ x)\ \uparrow V$
$\uparrow (W\ (\lambda x.M)\ V)$	$= \uparrow W\ \uparrow (\lambda x. \uparrow M)\ \uparrow V$
$\uparrow x$	$= x$
$\uparrow \lambda k.\lambda x.M$	$= \lambda k.\lambda x. \uparrow M$
$\uparrow k$	$= k$
$\uparrow \lambda x.M$	$= \lambda x. \uparrow M$

Figure 14: Projection of $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$ onto $\lambda^{f}_{\mathsf{CPS}}$

Remark 19 Note that, in $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$, $\uparrow M \longrightarrow_{\eta_{\mathsf{V}}2}\ M$, $\uparrow V \longrightarrow_{\eta_{\mathsf{V}}2}\ V$, $\uparrow K \longrightarrow_{\eta_{\mathsf{V}}2} K$ and if M, V, K are in $\lambda^{f}_{\mathsf{CPS}}$ then $\uparrow M = M$, $\uparrow V = V$, $\uparrow K = K$.

Theorem 20 (Galois connection from $\lambda^{f}_{\mathsf{CPS}}$ to $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$) *The identity $\mathsf{Id}_{\lambda^{f}_{CPS}}$ and the mapping $\uparrow$ form a Galois connection from λ^{f}_{CPS}, equipped with $\lambda^{f}_{CPS\beta\eta}$, to $\lambda^{\mathcal{F}}_{CPS}$, equipped with $\lambda^{\mathcal{F}}_{CPS\beta\eta}$, (and also with only $\lambda^{f}_{CPS\beta}$ and $\lambda^{\mathcal{F}}_{CPS\beta}$).*

Proof: Given Remark 19, it suffices to check the simulations.

- For the simulation of $\lambda^{\mathcal{F}}_{\mathsf{CPS}\beta\eta}$ by $\lambda^{f}_{\mathsf{CPS}\beta\eta}$ through $\uparrow$, we use a straightforward induction on the derivation of the reduction step, using the following fact:

$$\begin{array}{ll} \uparrow M\{x =\uparrow V\} = \uparrow M\{x = V\} & \uparrow M\{k =\uparrow K\} \longrightarrow^{*}_{\beta_{\mathsf{V}}1} \uparrow M\{k = K\} \\ \uparrow W\{x =\uparrow V\} = \uparrow W\{x = V\} & \uparrow W\{k =\uparrow K\} \longrightarrow^{*}_{\beta_{\mathsf{V}}1} \uparrow W\{k = K\} \\ \uparrow K'\{x =\uparrow V\} = \uparrow K'\{x = V\} & \uparrow K'\{k =\uparrow K\} \longrightarrow^{*}_{\beta_{\mathsf{V}}1} \uparrow K'\{k = K\} \end{array}$$

$$\begin{array}{l} \uparrow M\{k =\uparrow \lambda x.k\ x\} \longrightarrow^{*}_{\beta_{\mathsf{V}}1} \uparrow M \\ \uparrow W\{k =\uparrow \lambda x.k\ x\} \longrightarrow^{*}_{\beta_{\mathsf{V}}1} \uparrow W \\ \uparrow K'\{k =\uparrow \lambda x.k\ x\} \longrightarrow^{*}_{\beta_{\mathsf{V}}1} \uparrow K' \end{array}$$

- The fact that $\beta_{\mathsf{V}}1, \beta_{\mathsf{V}}2, \eta_{\mathsf{V}}1, \eta_{\mathsf{V}}2$ simulate $\beta_{\mathsf{V}}1, \beta_{\mathsf{V}}3, \eta_{\mathsf{V}}3$ through $\mathsf{Id}_{\lambda^{f}_{\mathsf{CPS}}}$ is straightforward.

□

Corollary 21 (Confluence of $\lambda^{f}_{\mathsf{CPS}}$) *$\lambda^{f}_{CPS\beta}$ and $\lambda^{f}_{CPS\beta\eta}$ are confluent.*

Proof: By Theorem 20 and Theorem 1. □

5.3 From $\lambda^{f}_{\mathsf{CPS}}$ to λ_{Q}

Definition 7 (The Fischer reverse translation)
We now encode $\lambda^{f}_{\mathsf{CPS}}$ into λ_{Q}.

$(k\ V)^{\mathsf{back}}$	$= [V^{\mathsf{back}}]$
$((\lambda x.M)\ V)^{\mathsf{back}}$	$= C_3(V^{\mathsf{back}}, x.M^{\mathsf{back}})$
$(y\ (\lambda x.M)\ V)^{\mathsf{back}}$	$= y(V^{\mathsf{back}}, x.M^{\mathsf{back}})$
$((\lambda k.\lambda z.N)\ (\lambda x.M)\ V)^{\mathsf{back}}$	$= C_3(\lambda z.N^{\mathsf{back}}, y.y(V^{\mathsf{back}}, x.M^{\mathsf{back}}))$
x^{back}	$= x$
$(\lambda k.\lambda x.M)^{\mathsf{back}}$	$= \lambda x.M^{\mathsf{back}}$

Lemma 22

1. $C_2(V^{\mathsf{back}}, x.W^{\mathsf{back}}) \longrightarrow^{*}_{\lambda_{\mathsf{Q}}}\ (W\{x = V\})^{\mathsf{back}}$ *and* $C_2(V^{\mathsf{back}}, x.M^{\mathsf{back}}) \longrightarrow^{*}_{\lambda_{\mathsf{Q}}}\ (M\{x = V\})^{\mathsf{back}}$.
2. $C_3(M^{\mathsf{back}}, x.N^{\mathsf{back}}) \longrightarrow^{*}_{\lambda_{\mathsf{Q}}}\ (M\{k = \lambda x.N\})^{\mathsf{back}}$ *(if $k \in \mathsf{FV}(M)$).*

Proof: By induction on W, M. □

Theorem 23 (Simulation of λ^f_{CPS} in λ_{Q})
The reduction relation $\longrightarrow_{\lambda^f_{\mathsf{CPS}\beta}}$ is simulated by $\longrightarrow_{\lambda_{\mathsf{Q}}}$ through $(\_)^{\mathsf{back}}$.

Proof: By induction on the derivation of the reduction step, using Lemma 22. □

Lemma 24 (Composition of the encodings)

1. *$V \longrightarrow^*_{\lambda_{\mathsf{C}\beta}} V^{\dagger\,\mathsf{back}}$ and $M \longrightarrow^*_{\lambda_{\mathsf{C}\beta}} (M\colon k)^{\mathsf{back}}$.*
2. *$V = V^{\mathsf{back}\,\dagger}$ and $M = M^{\mathsf{back}}\colon k$.*

Proof: By structural induction, using Lemma 22 for the first point. □

Now we can prove the following:

Theorem 25 (The refined Fischer translation is a reflection)
The refined Fischer translation and $(\_)^{\mathsf{back}}$ form a reflection in λ_{Q} of λ^f_{CPS} (equipped with $\lambda^f_{\mathsf{CPS}\beta}$).

Proof: This theorem is just the conjunction of Theorem 18, Theorem 23, and Lemma 24. □

Corollary 26 (Confluence of λ_{Q}-reductions) *λ_{Q} is confluent.*

Proof: By Theorem 25 and Theorem 1. □

6 Connection with Call-by-Value λ-calculus

We have established three connections:

- a reflection in λ_{Q} of λ^f_{CPS},
- a Galois connection from λ^f_{CPS} to $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$,
- a reflection in λ_{C} of $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$ (in section 3.4).

By composing the first two connections, we have a Galois connection from λ_{Q} to $\lambda^{\mathcal{F}}_{\mathsf{CPS}}$, and together with the last one, we have a a pre-Galois connection from λ_{Q} to λ_{C}. The compositions of these connections also form an equational correspondence between λ_{Q} to λ_{C}. These facts imply the following theorem:

Theorem 27 (Connections between λ_{Q} and λ_{C})
Let us write $V^\natural$ for $(\uparrow (V^{\mathcal{F}}))^{\mathsf{back}}$ and $M^\sharp$ for $(\uparrow (M\colon_{\mathcal{F}} k))^{\mathsf{back}}$.
Let us write $V^\flat$ for $(V^\dagger)^{\mathcal{F}\mathsf{back}}$ and $M^\flat$ for $(M\colon k)^{\mathcal{F}\mathsf{back}}$.

1. *For any terms M and N of λ_{C}, if $M \longrightarrow_{\lambda_{\mathsf{C}\beta}} N$ then $M^\sharp \longrightarrow^*_{\lambda_{\mathsf{Q}}} N^\sharp$.*
2. *For any terms M and N of λ_{Q}, if $M \longrightarrow_{\lambda_{\mathsf{Q}}} N$ then $M^\flat \longrightarrow^*_{\lambda_{\mathsf{C}\beta}} N^\flat$.*
3. *For any term M of λ_{C}, $M \longleftrightarrow^*_{\lambda_{\mathsf{C}\beta}} M^{\sharp\flat}$.*
4. *For any term M of λ_{Q}, $M \longrightarrow^*_{\lambda_{\mathsf{Q}}} M^{\flat\sharp}$.*

Proof: By composition of the reflections and Galois connection. □

Corollary 28 (Equational correspondence between λ_{Q} and λ_{C})

1. *For any terms M and N of λ_{Q}, $M \longleftrightarrow^*_{\lambda_{\mathsf{Q}}} N$ if and only if $M^\flat \longleftrightarrow^*_{\lambda_{\mathsf{C}\beta}} N^\flat$.*
2. *For any terms M and N of λ_{C}, $M \longleftrightarrow^*_{\lambda_{\mathsf{C}\beta}} N$ if and only if $M^\sharp \longleftrightarrow^*_{\lambda_{\mathsf{Q}}} N^\sharp$.*

We can give the composition of encodings explicitly. We have for instance the following theorem:

Theorem 29 (From λ_{Q} to λ_{C}) *The following equations hold, for the encoding from λ_{Q} to λ_{C}:*

$x^\natural$	$= x$
$(\lambda x.M)^\natural$	$= \lambda x.M^\sharp$
$V^\sharp$	$= [V^\natural]$
$(\mathsf{let}\ y = x\ V\ \mathsf{in}\ P)^\sharp$	$= x(V^\natural, y.P^\sharp)$
$(\mathsf{let}\ y = (\lambda x.M)\ V\ \mathsf{in}\ P)^\sharp$	$= C_3(\lambda x.M^\sharp, z.z(V^\natural, y.P^\sharp))$
$(\mathsf{let}\ z = V\ N\ \mathsf{in}\ P)^\sharp$	$= (\mathsf{let}\ y = N\ \mathsf{in}\ (\mathsf{let}\ z = V\ y\ \mathsf{in}\ P))^\sharp$ *if N is not a value*
$(\mathsf{let}\ z = M\ N\ \mathsf{in}\ P)^\sharp$	$= (\mathsf{let}\ x = M\ \mathsf{in}\ (\mathsf{let}\ z = x\ N\ \mathsf{in}\ P))^\sharp$ *if M is not a value*
$(\mathsf{let}\ z = (\mathsf{let}\ x = M\ \mathsf{in}\ N)\ \mathsf{in}\ P)^\sharp$	$= (\mathsf{let}\ x = M\ \mathsf{in}\ (\mathsf{let}\ z = N\ \mathsf{in}\ P))^\sharp$
$(\mathsf{let}\ y = V\ \mathsf{in}\ P)^\sharp$	$= C_3(V^\sharp, y.P^\sharp)$
$(M\ N)^\sharp$	$= (\mathsf{let}\ y = M\ N\ \mathsf{in}\ y)^\sharp$

Proof: By structural induction, unfolding the definition of the encodings on both sides of each equation. □

In fact, we could take this set of equalities as the definition of the direct encoding of λ_{C} into λ_{Q}. For that it suffices to check that there is a measure that makes this set of equations a well-founded definition. We now give this measure, given by an encoding of the terms of λ_{C} into first-order terms equipped with a *Lexicographic Path Ordering* (LPO) [15].

Definition 8 (An LPO for λ_{C}) We encode λ_{C} into the first-order syntax given by the following term constructors and their precedence relation:

$$\mathsf{ap}(\_,\_) \succ \mathsf{let}(\_,\_) \succ \mathsf{ii}(\_,\_) \succ \mathsf{i}(\_) \succ \star$$

The precedence generates a terminating LPO $\gg$ as defined in [15]. The encoding is given in Fig. 15. We can now consider the (terminating) relation induced by $\gg$ through the reverse relation of the encoding as a measure for λ_{C}.

$\overline{x}$	$= \star$	
$\overline{\lambda x.M}$	$= \mathsf{i}(\overline{M})$	
$\overline{\mathsf{let}\ x = M_1\ M_2\ \mathsf{in}\ N}$	$= \mathsf{let}(\mathsf{ii}(\overline{M_1}, \overline{M_2}), \overline{N})$	
$\overline{\mathsf{let}\ x = M\ \mathsf{in}\ N}$	$= \mathsf{let}(\overline{M}, \overline{N})$	otherwise
$\overline{M\ N}$	$= \mathsf{ap}(\overline{M}, \overline{N})$	

Figure 15: Encoding of λ_{C} into the first-order syntax

Remark 30 $\mathsf{let}(\overline{M}, \overline{N}) \gg \overline{\mathsf{let}\ x = M\ \mathsf{in}\ N}$ or $\mathsf{let}(\overline{M}, \overline{N}) = \overline{\mathsf{let}\ x = M\ \mathsf{in}\ N}$.

In the other direction, we have:

Theorem 31 (From λ_{C} to λ_{Q}) *The following properties hold, for the encoding from λ_{C} to λ_{Q}:*

$x^\flat$	$=$	x
$(\lambda x.M)^\flat$	$=$	$\lambda x.M^\flat$
$C_1(V, x.V')^\flat$	$=$	$V'^\flat\{x = V^\flat\}$
$[V]^\flat$	$=$	$V^\flat$
$(x(V, y.M))^\flat$	$=$	$\mathsf{let}\ y = x\ V^\flat\ \mathsf{in}\ M^\flat$
$C_3(N, x.M)^\flat$	$\underset{\lambda_{\mathsf{C}\beta}}{\overset{*}{\longleftarrow}}$	$\mathsf{let}\ x = N^\flat\ \mathsf{in}\ M^\flat$
$C_2(V, x.M)^\flat$	$=$	$M^\flat\{x = V^\flat\}$

Proof: By structural induction, unfolding the definition of the encodings on each side and using Lemma 10. □

Remark 32 Note from Theorem 31 and Theorem 29 that if M is a cut-free term of λ_{Q}, $M^{\flat\sharp} = M$. Also note that we do *not* have an equality in the penultimate line but only a reduction. This prevents the use of this theorem as a definition for the encoding from λ_{Q} to λ_{C}, although refining this case (with different sub-cases) could lead to a situation like that of Theorem 29 (with a measure to find for the set of equations to form a well-formed definition).

The connection between λ_{Q} and λ_{C} suggests to restricts λ_{C} by always requiring (a series of) application(s) to be explicitly given with a continuation, i.e. to refuse a term of λ_{C} such as $\lambda x.M_1\ M_2\ M_3$ but only accept $\lambda x.\mathsf{let}\ y = M_1\ M_2\ M_3\ \mathsf{in}\ y$. The refined Fisher translation of λ_{C}, on that particular fragment, directly has λ^f_{CPS} as its target calculus, and the equational correspondence between λ_{C} d λ_{Q} would then also become a pre-Galois connection from the former to the latter. The fragment is not stable under rule η_{let}, and corresponds to the terms of λ_{C} in some notion of η_{let}*-long normal form.*

This restriction can be formalised with the syntax of a calculus given in [6]:

$$\begin{array}{ll} M, N, P & ::= x \mid \lambda x.M \mid \mathsf{let}\ x = E\ \mathsf{in}\ M \\ E & ::= M \mid E\ M \end{array}$$

In fact, this calculus is introduced in [6] as a counterpart, in natural deduction, of a proof-term calculus for an unrestricted sequent calculus (i.e. not specific to the restriction of either LJT or LJQ). The calculus of [6] in natural deduction should thus allow the identification of CBN and CBV reductions as particular sub-reduction systems. Notions corresponding to the restrictions of LJT and LJQ should then capture both the traditional λ-calculus and (the η_{let}-long normal forms of) λ_{C}, respectively.

7 Conclusion

In this paper we investigated the call-by-value λ-calculus, and defined continuation-passing-style translations (and their refinements) in the style of Reynolds and Fischer. We have identified the target calculi and proved confluence of that of Fischer. We then presented Moggi's λ_{C}-calculus and proved that a decomposition of its main rule into two steps allowed the refined Fischer translation to form a reflection. Such a decomposition brings λ_{C} closer to the sequent calculus LJQ.

Indeed, we established new results about LJQ, in connection with Moggi's λ_{C}-calculus. We have introduced a proof-term syntax for LJQ, together with reduction rules expressing cut-elimination. Such a calculus, called λ_{Q}, can be considered independently from typing and still have a call-by-value semantics given by an adaptation of the Fischer CPS-translation. It relates to λ_{C} by two encodings that form both an equational correspondence and a pre-Galois

connection. In particular, the reductions in one calculus are simulated by those of the other.

As mentioned in section 4, we leave the proof of strong normalisation of λ_{Q} for another paper, since the result pertains to a typed framework and thus involves some more specific machinery (the typing system was given here to help the intuition about proof-terms).

Further work includes refining the encodings and/or cut-reduction system of λ_{Q}, so that the two calculi can be related by a Galois connection or a reflection.

Another promising direction for further work is given by the calculus from [6] in natural deduction that encompass both the traditional (CBN) λ-calculus and (a minor variant of) the CBV λ_{C}-calculus, with a very strong connection with sequent calculus as a whole (rather than either LJT or LJQ).

References

[1] Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *J. Logic Comput.*, 2(3):297–347, 1992.

[2] Vincent Danos, Jean-Baptiste Joinet, and Harold Schellinx. Lkq and lkt: sequent calculi for second order logic based upon dual linear decompositions of classical implication. In Jean-Yves Girard, Yves Lafont, and Laurent Regnier, editors, *Proc. of the Work. on Advances in Linear Logic*, volume 222 of *London Math. Soc. Lecture Note Ser.*, pages 211–224. Cambridge University Press, 1995.

[3] R Dyckhoff and C Urban. Strong normalization of Herbelin's explicit substitution calculus with substitution propagation. *J. Logic Comput.*, 13(5):689–706, 2003.

[4] Roy Dyckhoff and Stéphane Lengrand. **LJQ**, a strongly focused calculus for intuitionistic logic. In A. Beckmann, U. Berger, B. Loewe, and J. V. Tucker, editors, *Proc. of the 2nd Conf. on Computability in Europe (CiE'06)*, volume 3988 of *LNCS*, pages 173–185. Springer-Verlag, July 2006.

[5] Roy Dyckhoff and Luis Pinto. Cut-elimination and a permutation-free sequent calculus for intuitionistic logic. *Studia Logica*, 60(1):107–118, 1998.

[6] José Espírito Santo. Unity in structural proof theory and structural extensions of the λ-calculus. Manuscript, July 2005.

[7] Michael J. Fischer. Lambda calculus schemata. In *Proc. of the ACM Conf. on Proving Assertions about Programs*, pages 104–109. SIGPLAN Notices, Vol. 7, No 1 and SIGACT News, No 14, January 1972.

[8] Michael J. Fischer. Lambda-calculus schemata. *LISP and Symbolic Computation*, 6(3/4):259–288, December 1993.

[9] Thérèse Hardin. *Résultats de confluence pour les règles fortes de la logique combinatoire catégorique et liens avec les lambda-calculs.* Thèse de doctorat, Université de Paris VII, 1987.

[10] H. Herbelin. *Séquents qu'on calcule.* Thèse de doctorat, Université Paris VII, 1995.

[11] Hugo Herbelin. A lambda-calculus structure isomorphic to Gentzen-style sequent calculus structure. In Leszek Pacholski and Jerzy Tiuryn, editors, *Computer Science Logic, 8th Int. Work. , CSL '94*, volume 933 of *LNCS*, pages 61–75. Springer-Verlag, September 1994.

[12] J. Hudelmaier. *Bounds for Cut Elimination in Intuitionistic Logic.* PhD thesis, Universität Tübingen, 1989.

[13] Jörg Hudelmaier. An o(n log n)-space decision procedure for intuitionistic propositional logic. *J. Logic Comput.*, 3(1):63–75, 1993.

[14] J-.B. Joinet. *Étude de la Normalisation du Calcul des Séquents Classique à Travers la Logique Linéaire*. PhD thesis, University of Paris VII, 1993.

[15] Samuel Kamin and Jean-Jacques Lévy. Attempts for generalizing the recursive path orderings. Handwritten paper, University of Illinois, 1980.

[16] Austin Melton, David A. Schmidt, and George Strecker. Galois connections and computer science applications. In David Pitt, Samson Abramsky, Axel Poigné, and David Rydeheard, editors, *Proc. of a Tutorial and Work. on Category Theory and Computer Programming*, volume 240 of *LNCS*, pages 299–312, New York, NY, USA, November 1986. Springer-Verlag.

[17] Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. *Ann. Pure Appl. Logic*, 51:125–157, 1991.

[18] Eugenio Moggi. Computational lambda-calculus and monads. Report ECS-LFCS-88-66, University of Edinburgh, Edinburgh, Scotland, October 1988.

[19] Eugenio Moggi. Notions of computation and monads. *Inform. and Comput.*, 93:55–92, 1991.

[20] G. D. Plotkin. Call-by-name, call-by-value and the lambda-calculus. *Theoret. Comput. Sci.*, 1:125–159, 1975.

[21] John C. Reynolds. Definitional interpreters for higher-order programming languages. In *Proc. of the ACM annual Conf.* , pages 717–740, 1972.

[22] John C. Reynolds. The discoveries of continuations. *LISP and Symbolic Computation*, 6(3–4):233–247, 1993.

[23] John C. Reynolds. Definitional interpreters for higher-order programming languages. *Higher-Order and Symbolic Computation*, 11(4):363–397, 1998.

[24] Amr Sabry and Matthias Felleisen. Reasoning about programs in continuation-passing style. *Lisp Symb. Comput.*, 6(3-4):289–360, 1993.

[25] Amr Sabry and Philip Wadler. A reflection on call-by-value. *ACM Trans. Program. Lang. Syst.*, 19(6):916–941, 1997.

[26] Nikolaj Nikolaevic Vorob'ev. A new algorithm for derivability in the constructive propositional calculus. *Amer. Math. Soc. Transl.*, 94(2):37–71, 1970.