Axioms for Recursion in Call-by-Value
(Extended Abstract)

Masahito Hasegawa and Yoshihiko Kakutani

Research Institute for Mathematical Sciences, Kyoto University
{hassei,kakutani}@kurims.kyoto-u.ac.jp

Abstract. We propose an axiomatization of fixpoint operators in typed call-by-value programming languages, and give its justifications in two ways. First, it is shown to be sound and complete for the notion of uniform T-fixpoint operators of Simpson and Plotkin. Second, the axioms precisely account for Filinski's fixpoint operator derived from an iterator (infinite loop constructor) in the presence of first-class controls, provided that we define the uniformity principle on such an iterator via a notion of effect-freeness (centrality). We also investigate how these two results are related in terms of the underlying categorical models.

1 Introduction

While the equational theories of *fixpoint operators* in call-by-name programming languages and in domain theory have been extensively studied and now there are some canonical axiomatizations (including the *iteration theories* [1] and *Conway theories*, equivalently *traced cartesian categories* [9] – see [18] for the latest account), there seems no such widely-accepted result in the context of *call-by-value (cbv) programming languages*. In this paper we propose a candidate of such an axiomatization, which consists of three simple axioms.

A type-indexed family of closed values $\mathsf{fix}^{\mathsf{v}}_{\sigma\to\tau} : ((\sigma \to \tau) \to \sigma \to \tau) \to \sigma \to \tau$ is called a *stable uniform call-by-value fixpoint operator* if the following conditions are satisfied:

1. (cbv fixpoint) For any value $F : (\sigma \to \tau) \to \sigma \to \tau$
   $\mathsf{fix}^{\mathsf{v}}_{\sigma\to\tau} F \;=\; \lambda x^{\sigma}.F\,(\mathsf{fix}^{\mathsf{v}}_{\sigma\to\tau} F)\,x$
2. (stability) For any value $F : (\sigma \to \tau) \to \sigma \to \tau$
   $\mathsf{fix}^{\mathsf{v}}_{\sigma\to\tau} F \;=\; \mathsf{fix}^{\mathsf{v}}_{\sigma\to\tau}\,(\lambda f^{\sigma\to\tau}.\lambda x^{\sigma}.F\,f\,x)$
3. (uniformity) For values $F : (\sigma \to \tau) \to \sigma \to \tau$, $G : (\sigma' \to \tau') \to \sigma' \to \tau'$ and $H : (\sigma \to \tau) \to \sigma' \to \tau'$, if $H(\lambda x^{\sigma}.M\,x) = \lambda y^{\sigma'}.H\,M\,y$ holds for any $M : \sigma \to \tau$ (such an H is called *rigid*) and $H \circ F = G \circ H$ holds, then $H\,(\mathsf{fix}^{\mathsf{v}}_{\sigma\to\tau} F) \;=\; \mathsf{fix}^{\mathsf{v}}_{\sigma'\to\tau'} G$

The first axiom is known as the *call-by-value fixpoint equation*; the eta-expansion in the right-hand-side means that $\mathsf{fix}^{\mathsf{v}}_{\sigma\to\tau} F$ is equal to a value. The second axiom

F. Honsell and M. Miculan (Eds.): FOSSACS 2001, LNCS 2030, pp. 246–260, 2001.

says that, though the functionals F and $\lambda f.\lambda x.F\, f\, x$ may behave differently, their fixpoints, when applied to values, satisfy the same fixpoint equation and cannot be distinguished. The last axiom is a call-by-value variant of Plotkin's *uniformity principle*; here the *rigid functionals* (the word "rigid" was coined by Filinski in [4]) play the rôle of *strict functions* in the uniformity principle for the call-by-name fixpoint operators. Intuitively, a rigid functional uses its argument exactly once, and it does not matter whether the argument is evaluated beforehand or evaluated at its actual use. Our uniformity axiom can be justified by the fact that $H\,(\mathsf{fix}^{\mathsf{v}}_{\sigma\to\tau}\,F)$ satisfies the same fixpoint equation as $\mathsf{fix}^{\mathsf{v}}_{\sigma'\to\tau'}\,G$ when H is rigid and $H \circ F = G \circ H$ holds:

$$\begin{aligned} H\,(\mathsf{fix}^{\mathsf{v}}_{\sigma\to\tau}\,F) &= H\,(\lambda x^{\sigma}.F\,(\mathsf{fix}^{\mathsf{v}}_{\sigma\to\tau}\,F)\,x) && \text{cbv fixpoint equation for } \mathsf{fix}^{\mathsf{v}}_{\sigma\to\tau}\,F \\ &= \lambda y^{\sigma'}.H\,(F\,(\mathsf{fix}^{\mathsf{v}}_{\sigma\to\tau}\,F))\,y && H \text{ is rigid} \\ &= \lambda y^{\sigma'}.G\,(H\,(\mathsf{fix}^{\mathsf{v}}_{\sigma\to\tau}\,F))\,y && H \circ F = G \circ H \end{aligned}$$

We give two main results on these axioms.

1. The λ_c-*calculus* (*computational lambda calculus*) [12] with a stable uniform cbv fixpoint operator is sound and complete for the models based on the notion of *uniform T-fixpoint operators* of Simpson and Plotkin [18].
2. In the *call-by-value* $\lambda\mu$-*calculus* [17] (= the λ_c-calculus plus *first-class continuations*) there is a bijective correspondence between stable uniform cbv fixpoint operators and *uniform iterators*, via Filinski's construction of *recursion from iteration* [4].

In fact, we distill our axioms from the uniform T-fixpoint operators, so the first result is not an unexpected one. A surprise is the second one, in that the axioms precisely account for Filinski's cbv fixpoint operator derived from an *iterator* (infinite loop constructor) and first-class continuations, provided that we refine Filinski's notion of uniformity, for which the distinction between values and effect-free programs [19,7] is essential.

So here is an interesting coincidence of a category-theoretic axiomatics (of Simpson and Plotkin) with a program construction (of Filinski). However, we also show that, after sorting out the underlying categorical semantics, Filinski's construction combined with the *Continuation-Passing Style (CPS) translation* can be understood within the abstract setting of Simpson and Plotkin.

Construction of this paper. In Section 2 we recall the λ_c-calculus and the call-by-value $\lambda\mu$-calculus, which will be used as our working languages in this paper. Section 3 demonstrates how our axioms are used for establishing the Filinski's correspondence between recursion and iteration (which can be seen as a syntactic proof of the second main result). Up to this section, all results are presented in an entirely syntactic manner. In Section 4 we start to look at the semantic counterpart of our axiomatization, by recalling the categorical models of the λ_c-calculus and the call-by-value $\lambda\mu$-calculus. We then recall the notion of uniform T-fixpoint operators on these models in Section 5, and explain how

our axioms are distilled from the uniform T-fixpoint operators (the first main result). In Section 6, we specialise the result in the previous section to the models of the call-by-value $\lambda\mu$-calculus, and give a semantic proof of the second main result. Section 7 gives some concluding remarks.

2 The Call-by-Value Calculi

The λ_c-calculus (computational lambda calculus) [12], an improvement of the call-by-value λ-calculus [14], is sound and complete for

1. categorical models based on *strong monads* [12]
2. Continuation-Passing Style translation into the $\lambda\beta\eta$-calculus [16]

and has been proved useful for reasoning about call-by-value programs. In particular, it can be seen as the theoretical backbone of (the typed version of) the theory of *A-normal forms* [6], which enables us to optimise call-by-value programs directly without performing the CPS translation.

For these reasons, we take the λ_c-calculus as a basic calculus for typed call-by-value programming languages. We also use an extension of the λ_c-calculus with first-class controls, called the call-by-value $\lambda\mu$-calculus, for which the soundness and completeness results mentioned above have been extended by Selinger [17].

2.1 The λ_c-Calculus

The syntax, typing rules and axioms on the well-typed terms of the λ_c-calculus are summarised in Figure 1. The types, terms and typing judgements are those of the standard simply typed lambda calculus (including the unit $\top$ and binary products $\times$). c^σ ranges over the constants of type σ. As an abbreviation, we write $\mathsf{let}\ x^\sigma\ \mathsf{be}\ M\ \mathsf{in}\ N$ for $(\lambda x^\sigma.N)M$. $\mathrm{FV}(M)$ denotes the set of free variables in M. The crucial point is that we have the notion of values, and the axioms are designed so that the above-mentioned completeness results hold. Below we may call a term a value if it is provably equal to a value defined by the grammar.

Centre and focus. In call-by-value languages, we often regard values as representing effect-free (finished or suspended) computation. While this intuition is valid, the converse may not always be justified; in fact, the answer depends on the computational effects under consideration [7]. In a λ_c-theory (where we may have additional constructs and axioms), we say that a term $M : \sigma$ is *central* if it commutes with any other computational effect, that is,

$$\mathsf{let}\ x^\sigma\ \mathsf{be}\ M\ \mathsf{in}\ \mathsf{let}\ y^\tau\ \mathsf{be}\ N\ \mathsf{in}\ L\ =\ \mathsf{let}\ y^\tau\ \mathsf{be}\ N\ \mathsf{in}\ \mathsf{let}\ x^\sigma\ \mathsf{be}\ M\ \mathsf{in}\ L\ :\ \theta$$

holds for any $N : \tau$ and $L : \theta$, where x and y are not free in M and N. In addition, we say that $M : \sigma$ is *focal* if it is central and moreover *copyable* and *discardable*, i.e., $\mathsf{let}\ x^\sigma\ \mathsf{be}\ M\ \mathsf{in}\ \langle x, x\rangle = \langle M, M\rangle : \sigma \times \sigma$ and $\mathsf{let}\ x^\sigma\ \mathsf{be}\ M\ \mathsf{in}\ * = * : \top$ hold. It is worth emphasising that a value is always focal, but the converse is not true. A detailed analysis of these concepts in several λ_c-theories is found in [7]; see also discussions in Section 4.

Types $\sigma, \tau \quad ::= b \mid \sigma \to \tau \mid \top \mid \sigma \times \tau$ where b ranges over base types
Terms $M, N ::= x \mid c^\sigma \mid \lambda x^\sigma.M \mid M\,N \mid * \mid \langle M, N\rangle \mid \pi_1\,M \mid \pi_2\,M$
Values $V, U \quad ::= x \mid c^\sigma \mid \lambda x^\sigma.M \mid * \mid \langle V, U\rangle \mid \pi_1\,V \mid \pi_2\,V$

Typing Rules:

$$\frac{}{\Gamma \vdash x : \sigma}\ x{:}\sigma \in \Gamma \qquad \frac{}{\Gamma \vdash c^\sigma : \sigma} \qquad \frac{\Gamma, x:\sigma \vdash M : \tau}{\Gamma \vdash \lambda x^\sigma.M : \sigma \to \tau} \qquad \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash M\,N : \tau}$$

$$\frac{}{\Gamma \vdash * : \top} \qquad \frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash N : \tau}{\Gamma \vdash \langle M, N\rangle : \sigma \times \tau} \qquad \frac{\Gamma \vdash M : \sigma \times \tau}{\Gamma \vdash \pi_1\,M : \sigma} \qquad \frac{\Gamma \vdash M : \sigma \times \tau}{\Gamma \vdash \pi_2\,M : \tau}$$

Axioms:

$\mathsf{let}\ x^\sigma\ \mathsf{be}\ V\ \mathsf{in}\ M$	$= M[V/x]$	
$\lambda x^\sigma.V\,x$	$= V$	$(x \notin \mathrm{FV}(V))$
V	$= *$	$(V : \top)$
$\pi_i\langle V_1, V_2\rangle$	$= V_i$	
$\langle \pi_1\,V, \pi_2\,V\rangle$	$= V$	
$\mathsf{let}\ x^\sigma\ \mathsf{be}\ M\ \mathsf{in}\ x$	$= M$	
$\mathsf{let}\ y^\tau\ \mathsf{be}\ (\mathsf{let}\ x^\sigma\ \mathsf{be}\ L\ \mathsf{in}\ M)\ \mathsf{in}\ N$	$= \mathsf{let}\ x^\sigma\ \mathsf{be}\ L\ \mathsf{in}\ \mathsf{let}\ y^\tau\ \mathsf{be}\ M\ \mathsf{in}\ N$	$(x \notin \mathrm{FV}(N))$
$M\,N$	$= \mathsf{let}\ f^{\sigma\to\tau}\ \mathsf{be}\ M\ \mathsf{in}\ \mathsf{let}\ x^\sigma\ \mathsf{be}\ N\ \mathsf{in}\ f\,x$	$(M : \sigma \to \tau, N : \sigma)$
$\langle M, N\rangle$	$= \mathsf{let}\ x^\sigma\ \mathsf{be}\ M\ \mathsf{in}\ \mathsf{let}\ y^\tau\ \mathsf{be}\ N\ \mathsf{in}\ \langle x, y\rangle$	$(M : \sigma, N : \tau)$
$\pi_i\,M$	$= \mathsf{let}\ x^{\sigma\times\tau}\ \mathsf{be}\ M\ \mathsf{in}\ \pi_i\,x$	$(M : \sigma \times \tau)$

Fig. 1. The λ_c-calculus

2.2 The Call-by-Value $\lambda\mu$-Calculus

Our call-by-value $\lambda\mu$-calculus, summarised in Figure 2, is the version due to Selinger [17]. We regard it as an extension of the λ_c-calculus with first-class continuations and sum types (the empty type $\bot$ and binary sums $+$). We write $\neg\sigma$ for the type $\sigma \to \bot$ ("negative type").

Remark 1. We have chosen the cbv $\lambda\mu$-calculus as our working language firstly because we intend the results in this paper to be compatible with the duality result of the second author [11] (see Section 7) which is based on Selinger's work on the $\lambda\mu$-calculus [17], and secondly because it has a well-established categorical semantics, again thanks to Selinger. However our results are not specific to the $\lambda\mu$-calculus; they apply also to any other language with similar semantics – for example, we could have used Hofmann's axiomatization of control operators [10]. Also, strictly speaking, the inclusion of sum types (coproducts) is not necessary in the main development of this paper, though they enable us to describe iterators more naturally (as general feedback operators, see Remark 2 in Section 3) and are also used in some principles on iterators like diagonal property (see Section 7), and crucially needed for the duality result in [17,11].

The typing judgements take the form $\Gamma \vdash M : \sigma \mid \Delta$ where $\Delta = \alpha_1 : \tau_1, \ldots, \alpha_n : \sigma_n$ is a sequence of *names* (ranged over by α, β,...) with their types. A judgement $x_1 : \sigma_1, \ldots, x_m : \sigma_m \vdash M : \tau \mid \alpha_1 : \tau_1, \ldots, \alpha_n : \tau_n$ represents

$$
\begin{array}{lll}
\text{Types } \sigma,\tau & ::= \cdots \mid \bot \mid \sigma+\tau & \\
\text{Terms } M,N & ::= \cdots \mid [\alpha]M \mid \mu\alpha^{\sigma}.M \mid [\alpha,\beta]M \mid \mu(\alpha^{\sigma},\beta^{\tau}).M & \\
\text{Values } V,U & ::= \cdots \mid \mu(\alpha^{\sigma},\beta^{\tau}).[\alpha]V \mid \mu(\alpha^{\sigma},\beta^{\tau}).[\beta]V & \text{where } \alpha,\beta \notin \mathrm{FN}(V)
\end{array}
$$

Additional Typing Rules:

$$
\frac{\Gamma \vdash M : \sigma \mid \Delta}{\Gamma \vdash [\alpha]M : \bot \mid \Delta}\ \alpha{:}\sigma \in \Delta
\qquad
\frac{\Gamma \vdash M : \bot \mid \alpha : \sigma, \Delta}{\Gamma \vdash \mu\alpha^{\sigma}.M : \sigma \mid \Delta}
$$

$$
\frac{\Gamma \vdash M : \sigma+\tau \mid \Delta}{\Gamma \vdash [\alpha,\beta]M : \bot \mid \Delta}\ \alpha{:}\sigma,\beta{:}\tau \in \Delta
\qquad
\frac{\Gamma \vdash M : \bot \mid \alpha : \sigma, \beta : \tau, \Delta}{\Gamma \vdash \mu(\alpha^{\sigma},\beta^{\tau}).M : \sigma+\tau \mid \Delta}
$$

Additional Axioms:

$$
\begin{array}{lll}
V\,(\mu\alpha^{\sigma}.M) & = \mu\beta^{\tau}.M[[\beta](V(-))/[\alpha](-)] & (V : \sigma \to \tau) \\
[\alpha'](\mu\alpha^{\sigma}.M) & = M[\alpha'/\alpha] & \\
\mu\alpha^{\sigma}.[\alpha]M & = M & (\alpha \notin \mathrm{FN}(M)) \\
[\alpha',\beta'](\mu(\alpha^{\sigma},\beta^{\tau}).M) & = M[\alpha'/\alpha,\beta'/\beta] & \\
\mu(\alpha^{\sigma},\beta^{\tau}).[\alpha,\beta]M & = M & (\alpha,\beta \notin \mathrm{FN}(M)) \\
[\alpha]M & = M & (M : \bot) \\
[\alpha]M & = \mathsf{let}\ x^{\sigma}\ \mathsf{be}\ M\ \mathsf{in}\ [\alpha]x & (M : \sigma) \\
[\alpha,\beta]M & = \mathsf{let}\ x^{\sigma+\tau}\ \mathsf{be}\ M\ \mathsf{in}\ [\alpha,\beta]x & (M : \sigma+\tau)
\end{array}
$$

Fig. 2. The call-by-value $\lambda\mu$-calculus

a well-typed term M with at most m *free variables* $x_1,\ldots,x_m$ and n *free names* $\alpha_1,\ldots,\alpha_n$. We write $\mathrm{FN}(M)$ for the set of free names in M. In this judgement, M can be thought as a proof of the sequent $\sigma_1,\ldots,\sigma_m \vdash \tau, \tau_1,\ldots,\tau_n$ or the proposition $(\sigma_1 \wedge \ldots \wedge \sigma_m) \to (\tau \vee \tau_1 \vee \ldots \vee \tau_n)$ in the classical propositional logic. Among the additional axioms, the first one involves the *mixed substitution* $M[C(-)/[\alpha](-)]$ for a term M, a context $C(-)$ and a name α, which is the result of recursively replacing any subterm of the form $[\alpha]N$ by $C(N)$ and any subterm of the form $[\alpha_1,\alpha_2]N$ (with $\alpha=\alpha_1$ or $\alpha=\alpha_2$) by $C(\mu\alpha.[\alpha_1,\alpha_2]N)$. See [17] for further details on these syntactic conventions.

Centre and focus. In the presence of first-class controls, central and focal terms coincide [19,17], and enjoy a simple characterisation (*thunkability* [19]).

Lemma 1. *In a cbv $\lambda\mu$-theory, the following conditions on a term $M : \sigma$ are equivalent.*

1. *M is central.*
2. *M is focal.*
3. *(thunkability)* $\mathsf{let}\ x^{\sigma}\ \mathsf{be}\ M\ \mathsf{in}\ \lambda k^{\neg\sigma}.k\,x = \lambda k^{\neg\sigma}.k\,M : \neg\neg\sigma$ *holds.*

We also note that central terms and values agree at function types [17].

Lemma 2. *In a cbv $\lambda\mu$-theory, a term $M : \sigma \to \tau$ is central if and only if it is a value, i.e., $M = \lambda x^{\sigma}.M\,x$ holds.*

3 Recursion from Iteration

For grasping the rôle of our axioms, it is best to look at the actual construction in the second main result: the correspondence of recursors and iterators under the presence of first-class continuations due to Filinski [4]. For ease of presentation, we write $g \circ f$ for the composition $\lambda x.g\,(f\,x)$ of values f and g, and id_σ for $\lambda x^\sigma.x$.

A type-indexed family of closed values $\mathsf{loop}_\sigma : (\sigma \to \sigma) \to \neg\sigma$ is called a *uniform iterator* if the following conditions are satisfied:

1. (iteration) For any value $f : \sigma \to \sigma$, $\quad \mathsf{loop}_\sigma\, f \;=\; \lambda x^\sigma.\mathsf{loop}_\sigma\, f\,(f\,x)$
2. (uniformity) For values $f : \sigma \to \sigma$, $g : \sigma' \to \sigma'$ and $h : \sigma \to \sigma'$, if h is *total* and $h \circ f = g \circ h$ holds, then $\;(\mathsf{loop}_{\sigma'}\, g) \circ h = \mathsf{loop}_\sigma\, f$

where a value $h : \sigma \to \tau$ is called *total* if $h\,v : \tau$ is central (see Section 2) for any value $v : \sigma$. The word "total" is due to Filinski [4], though in his original definition $h\,v$ is asked to be a *value* rather than a central term.[1]

Remark 2. The expressive power of an iterator is not so weak, as we can derive a general *feedback operator* $\mathsf{feedback}_{\sigma,\tau} : (\sigma \to \sigma + \tau) \to \sigma \to \tau$ from an iterator using sums and first-class controls, which satisfies (with a syntax sugar for sums) $\mathsf{feedback}_{\sigma,\tau}\, f\, a \;=\; \mathsf{case}\; f\, a\; \mathsf{of}\; (\mathsf{in}_1\, x^\sigma \Rightarrow \mathsf{feedback}_{\sigma,\tau}\, f\, x \,|\, \mathsf{in}_2\, y^\tau \Rightarrow y)$ for values $f : \sigma \to \sigma + \tau$ and $a : \sigma$.

Surprisingly, in the presence of first-class continuations, there is a bijective correspondence between the stable uniform cbv fixpoint operators and the uniform iterators. We recall the construction which is essentially the same as that in [4]. Sample codes are found in Figure 3 (in SML/NJ [8]) and 4 (in the cbv $\lambda\mu$-calculus).

The construction is divided into two parts. For the first part, we introduce a pair of "inside-out" (contravariant) constructions

$$\begin{cases} \mathsf{step}_{\sigma,\tau} \;:\; (\neg\tau \to \neg\sigma) \to \sigma \to \tau \\ \mathsf{pets}_{\sigma,\tau} = \lambda f^{\sigma\to\tau}.\lambda k^{\neg\tau}.\lambda x^\sigma.k\,(f\,x) \;:\; (\sigma \to \tau) \to \neg\tau \to \neg\sigma \end{cases}$$

so that $\mathsf{step}_{\sigma,\tau} \circ \mathsf{pets}_{\sigma,\tau} = \mathsf{id}_{\sigma\to\tau}$ and $\mathsf{pets}_{\sigma,\tau} \circ \mathsf{step}_{\sigma,\tau} = \lambda F^{\neg\tau\to\neg\sigma}.\lambda k^{\neg\tau}.\lambda x^\sigma.F\,k\,x$ hold; here we need first-class continuations to implement $\mathsf{step}_{\sigma,\tau}$. We are then able to see that, if loop is a uniform iterator, the composition

$$\mathsf{loop}_\sigma \circ \mathsf{step}_{\sigma,\sigma} \;:\; (\neg\sigma \to \neg\sigma) \to \neg\sigma$$

yields a stable uniform fixpoint operator restricted on the negative types $\neg\sigma$. In particular, the cbv fixpoint axiom is verified as (by noting the equation $k^{\neg\tau}\,(\mathsf{step}_{\sigma,\tau}\, F^{\neg\tau\to\neg\sigma}\, x^\sigma) \;=\; F\,k\,x$)

$$\begin{aligned} (\mathsf{loop}_\sigma \circ \mathsf{step}_{\sigma,\sigma})\, F &= \mathsf{loop}_\sigma\,(\mathsf{step}_{\sigma,\sigma}\, F) \\ &= \lambda x^\sigma.\mathsf{loop}_\sigma\,(\mathsf{step}_{\sigma,\sigma}\, F)\,(\mathsf{step}_{\sigma,\sigma}\, F\, x) \\ &= \lambda x^\sigma.F\,(\mathsf{loop}_\sigma\,(\mathsf{step}_{\sigma,\sigma}\, F))\, x \\ &= \lambda x^\sigma.F\,((\mathsf{loop}_\sigma \circ \mathsf{step}_{\sigma,\sigma})\, F)\, x \end{aligned}$$

[1] Our use of the word "total" can be misleading, as there is a more general and perhaps more sensible notion of "totality" used in Thielecke's analysis [20]. However in this paper we put our priority on the compatibility with Filinski's development in [4].

Conversely, if $\mathsf{fix}^{\mathsf{v}}$ is a stable uniform fixpoint operator,

$$\mathsf{fix}^{\mathsf{v}}_{\neg\sigma} \circ \mathsf{pets}_{\sigma,\sigma} \ : \ (\sigma \to \sigma) \to \neg\sigma$$

gives a uniform iterator:

$$\begin{aligned}(\mathsf{fix}^{\mathsf{v}}_{\neg\sigma} \circ \mathsf{pets}_{\sigma,\sigma})\, f &= \mathsf{fix}^{\mathsf{v}}_{\neg\sigma}\,(\mathsf{pets}_{\sigma,\sigma}\, f)\\ &= \lambda x^{\sigma}.(\mathsf{pets}_{\sigma,\sigma}\, f)\,(\mathsf{fix}^{\mathsf{v}}_{\neg\sigma}\,(\mathsf{pets}_{\sigma,\sigma}\, f))\, x\\ &= \lambda x^{\sigma}.(\lambda k^{\neg\sigma}.\lambda y^{\sigma}.k\,(f\,y))\,(\mathsf{fix}^{\mathsf{v}}_{\neg\sigma}\,(\mathsf{pets}_{\sigma,\sigma}\, f))\, x\\ &= \lambda x^{\sigma}.(\mathsf{fix}^{\mathsf{v}}_{\neg\sigma}\,(\mathsf{pets}_{\sigma,\sigma}\, f))\,(f\, x)\\ &= \lambda x^{\sigma}.(\mathsf{fix}^{\mathsf{v}}_{\neg\sigma} \circ \mathsf{pets}_{\sigma,\sigma})\, f\,(f\, x)\end{aligned}$$

One direction of the bijectivity of these constructions is guaranteed by the stability axiom (while the other direction follows from $\mathsf{step}_{\sigma,\sigma} \circ \mathsf{pets}_{\sigma,\sigma} = \mathsf{id}_{\sigma\to\sigma}$):

$$\mathsf{fix}^{\mathsf{v}}_{\neg\sigma} \circ \mathsf{pets}_{\sigma,\sigma} \circ \mathsf{step}_{\sigma,\sigma} = \lambda F.\mathsf{fix}^{\mathsf{v}}_{\neg\sigma}(\lambda k.\lambda x.F\,k\,x) = \lambda F.\mathsf{fix}^{\mathsf{v}}_{\neg\sigma}\, F = \mathsf{fix}^{\mathsf{v}}_{\neg\sigma}$$

We note that step sends a rigid function to a total one, while pets sends a total function to a rigid one, and moreover they are (contravariantly) functorial. These facts imply that the two notions of uniformity for recursors and iterators are in perfect harmony.

The second part is to reduce fixpoints on an arrow type $\sigma \to \tau$ to those on a negative type $\neg(\sigma \times \neg\tau)$. This is possible because we can implement an isomorphism (again using first-class continuations)

$$\mathsf{switch}_{\sigma,\tau} \ : \ \neg(\sigma \times \neg\tau) \xrightarrow{\simeq} \sigma \to \tau$$

which is rigid ($\mathsf{switch}_{\sigma,\tau}\,(\lambda x^{\sigma}.M\, x) = \lambda y^{\sigma\times\neg\tau}.\,\mathsf{switch}_{\sigma,\tau}\, M\, y$ holds) and we have

$$\mathsf{fix}^{\mathsf{v}}_{\sigma\to\tau}\, F \ = \ \mathsf{switch}_{\sigma,\tau}\,(\mathsf{fix}^{\mathsf{v}}_{\neg(\sigma\times\neg\tau)}\,(\mathsf{switch}^{-1}_{\sigma,\tau} \circ F \circ \mathsf{switch}_{\sigma,\tau}))$$

by the uniformity ($\mathsf{switch}_{\sigma,\tau} \circ (\mathsf{switch}^{-1}_{\sigma,\tau} \circ F \circ \mathsf{switch}_{\sigma,\tau}) = F \circ \mathsf{switch}_{\sigma,\tau}$). So we conclude that, under the presence of first-class continuations, stable uniform cbv fixpoint operators are precisely those derived from uniform iterators, and vice versa:

$$\begin{aligned}\mathsf{fix}^{\mathsf{v}}_{\sigma\to\tau}\, F \ &= \ \mathsf{switch}_{\sigma,\tau}(\mathsf{loop}_{\sigma\times\neg\tau}(\mathsf{step}_{\sigma\times\neg\tau,\sigma\times\neg\tau}(\mathsf{switch}^{-1}_{\sigma,\tau} \circ F \circ \mathsf{switch}_{\sigma,\tau})))\\ \mathsf{loop}_{\sigma}\, f \ &= \ \mathsf{fix}^{\mathsf{v}}_{\neg\sigma}(\mathsf{pets}_{\sigma,\sigma}\, f)\end{aligned}$$

Behind syntax. As noted by Filinski, the CPS-transform of an iterator is a usual (call-by-name) fixpoint operator on the types of the form R^A in the target $\lambda\beta\eta$-calculus, where R is the answer type. If we let T be the continuation monad $R^{R^{(-)}}$, then the uniform T-fixpoint operator of Simpson and Plotkin [18] precisely amounts to the uniform fixpoint operator on the types R^A.

Since our first main result (Section 5, Theorem 1) is that the stable uniform cbv fixpoint operator is sound and complete for uniform T-fixpoint operators, it turns out that Filinski's construction combined with the CPS translation can be regarded as a consequence of the general categorical axiomatics. By specialising it to the setting with a continuation monad, we obtain a semantic version of the *recursion from iteration* construction (Section 6, Theorem 2 and 3).

```
(* an empty type "bot" with an initial map "abort" A : bot -> 'a *)
datatype bot = VOID of bot;
fun A (VOID v) = A v;
(* the C operator, C : (('a -> bot) -> bot) -> 'a *)
fun C f = SMLofNJ.Cont.callcc
            (fn k => A (f (fn x => (SMLofNJ.Cont.throw k x) : bot)));

(* basic combinators *)
fun step F x = C (fn k => F k x);
fun pets f k x = k (f x) : bot;
fun switch l x = C (fn q => l (x,q));
fun switch_inv f (x, k) = k (f x) : bot;
(* step : (('a -> bot) -> 'b -> bot) -> 'b -> 'a
   pets : ('a -> 'b) -> ('b -> bot) -> 'a -> bot
   switch : ('a * ('b -> bot) -> bot) -> 'a -> 'b
   switch_inv : ('a -> 'b) -> 'a * ('b -> bot) -> bot *)

(* an iterator, loop : ('a -> 'a) -> 'a -> bot *)
fun loop f x = loop f (f x) : bot;

(* recursion from iteration *)
fun fix F = switch (loop (step (switch_inv o F o switch)));
(* fix : (('a -> 'b) -> 'a -> 'b) -> 'a -> 'b *)
```

Fig. 3. Coding in SML/NJ (versions based on SML '97)

4 Categorical Semantics

4.1 Models of the λ_c-Calculus

Let $\mathcal{C}$ be a category with finite products and a strong monad $T = (T, \eta, \mu, \theta)$. We write $\mathcal{C}_T$ for the Kleisli category of T, and $J : \mathcal{C} \to \mathcal{C}_T$ for the associated left adjoint functor. We assume that $\mathcal{C}$ has Kleisli exponentials, i.e., for every X in $\mathcal{C}$ the functor $J((-) \times X) : \mathcal{C} \to \mathcal{C}_T$ has a right adjoint $X \Rightarrow (-) : \mathcal{C}_T \to \mathcal{C}$. This gives the structure for modelling computational lambda calculus [12]. Following Moggi, we call such a structure a *computational model*.

$$
\begin{aligned}
\mathsf{step}_{\sigma,\tau} &= \lambda F^{\neg\tau\to\neg\sigma}.\lambda x^{\sigma}.\mu\beta^{\tau}.F(\lambda y^{\tau}.[\beta]y)x &&: (\neg\tau \to \neg\sigma) \to \sigma \to \tau \\
\mathsf{pets}_{\sigma,\tau} &= \lambda f^{\sigma\to\tau}.\lambda k^{\neg\tau}.\lambda x^{\sigma}.k(f\,x) &&: (\sigma \to \tau) \to \neg\tau \to \neg\sigma \\
\mathsf{switch}_{\sigma,\tau} &= \lambda l^{\neg(\sigma\times\neg\tau)}.\lambda x^{\sigma}.\mu\beta^{\tau}.l\langle x, \lambda y^{\tau}.[\beta]y\rangle &&: \neg(\sigma \times \neg\tau) \to \sigma \to \tau \\
\mathsf{switch}^{-1}_{\sigma,\tau} &= \lambda f^{\sigma\to\tau}.\lambda\langle x^{\sigma}, k^{\neg\tau}\rangle.k(f\,x) &&: (\sigma \to \tau) \to \neg(\sigma \times \neg\tau)
\end{aligned}
$$

Fig. 4. Coding in the call-by-value $\lambda\mu$-calculus

4.2 Models of the Call-by-Value $\lambda\mu$-Calculus

Let $\mathcal{C}$ be a distributive category, i.e., a category with finite products and coproducts so that $(-) \times A : \mathcal{C} \to \mathcal{C}$ preserves finite coproducts for each A. We call an object R a *response object* if there exists an exponential R^A for each A, i.e., $\mathcal{C}(- \times A, R) \simeq \mathcal{C}(-, R^A)$ holds. Given such a structure, we can model the cbv $\lambda\mu$-calculus in the Kleisli category $\mathcal{C}_T$ of the strong monad $T = R^{R^{(-)}}$ [17]. Following Selinger, we call $\mathcal{C}$ a *response category* and the Kleisli category $\mathcal{C}_T$ a *category of continuations* and write $R^{\mathcal{C}}$ for $\mathcal{C}_T$ (though in [17] a category of continuations means the opposite of $R^{\mathcal{C}}$).

4.3 Centre and Focus

We have already seen the notion of centre and focus in the λ_c-calculus and the cbv $\lambda\mu$-calculus in a syntactic form (Section 2). However, these concepts originally arose from the analysis on the category-theoretic models given as above. Following the discovery of the *premonoidal structure* on the Kleisli category part $\mathcal{C}_T$ ($R^{\mathcal{C}}$) of these models [15], Thielecke [19] proposed a *direct* axiomatization of $R^{\mathcal{C}}$ not depending on the base category $\mathcal{C}$ (which may be seen as a chosen category of "values") but on the subcategory of "effect-free" morphisms of $R^{\mathcal{C}}$, which is the *focus* (equivalently *centre*) of $R^{\mathcal{C}}$. Führmann [7] carries out further study on models of the λ_c-calculus along this line.

For lack of space we do not describe the details of these analyses. However, we will soon see that these concepts naturally arise in our analysis of the uniformity principles for recursors and iterators. In particular, a total value $h : \sigma \to \tau$ (equivalently the term $x : \sigma \vdash h\, x : \tau$) precisely corresponds to the central morphisms in the semantic models. In the case of the models of the cbv $\lambda\mu$-calculus, the centre can be characterised in terms of the category of algebras, for which our uniformity principles are defined; that is, we have

Proposition 1. *$f \in R^{\mathcal{C}}(A, B) \simeq \mathcal{C}(R^B, R^A)$ is central if and only if its counterpart in $\mathcal{C}$ is an algebra morphism from the canonical algebra structure on R^B (see Section 6) to that on R^A.*

We note that this result has been observed in various forms in [19,17,7].

5 Uniform T-Fixpoint Operators

In this section we shall consider a computational model with the base category $\mathcal{C}$ and a strong monad T.

Definition 1. *[18] A* T-fixpoint operator on $\mathcal{C}$ *is a family of functions*

$$(-)^* : \mathcal{C}(TX, TX) \to \mathcal{C}(1, TX)$$

such that, for any $f : TX \to TX$, $f \circ f^* = f^*$ holds. It is called uniform *if, for any $f : TX \to TX$, $g : TY \to TY$ and $h : TX \to TY$, $h \circ \mu = \mu \circ Th$ and $g \circ h = h \circ f$ imply $g^* = h \circ f^*$.*

Thus a T-fixpoint operator is given as a fixpoint operator restricted on the objects of the form TX. However, this is sufficient to model a call-by-value fixpoint operator. To see this, suppose that we are given an object A with an arrow $\alpha : TA \to A$ so that $\alpha \circ \eta = id$ (in fact it is more natural to ask (A, α) to be a T-algebra, see Proposition 2 below). Given $f : A \to A$, we have $\alpha \circ (\eta \circ f \circ \alpha)^* : 1 \to A$ and

$$\begin{aligned} \alpha \circ (\eta \circ f \circ \alpha)^* &= \alpha \circ \eta \circ f \circ \alpha \circ (\eta \circ f \circ \alpha)^* \\ &= f \circ \alpha \circ (\eta \circ f \circ \alpha)^* \end{aligned}$$

Therefore we can extend $(-)^*$ to be a fixpoint operator on A.

Definition 2. *[18] Suppose that $\mathcal{S}$ and $\mathcal{D}$ are categories with finite products and the same objects, and $I : \mathcal{S} \to \mathcal{D}$ is a functor which strictly preserves finite products and is the identity on objects. A* parameterized fixpoint operator on $\mathcal{D}$ *is a family of functions $(-)^\dagger : \mathcal{D}(X \times A, A) \to \mathcal{D}(X, A)$ which is natural in X and satisfies $f^\dagger = f \circ \langle id_X, f^\dagger \rangle$. It is* parametrically uniform with respect to $I : \mathcal{S} \to \mathcal{D}$ *if, for any $f : X \times A \to A$, $g : X \times B \to B$ in $\mathcal{D}$ and $h : A \to B$ in $\mathcal{S}$, $Ih \circ f = g \circ (id_X \times Ih)$ implies $g^\dagger = Ih \circ f^\dagger$.*

Proposition 2. *[18] Let $\mathcal{C}^T$ be the category of T-algebras and algebra morphisms. Let $\mathcal{D}$ be the category whose objects are T-algebras and hom-sets are given by $\mathcal{D}((A, \alpha), (B, \beta)) = \mathcal{C}(A, B)$, and let $I : \mathcal{C}^T \to \mathcal{D}$ be the inclusion. Then a uniform T-fixpoint operator on $\mathcal{C}$ induces a parametrically uniform parameterized fixpoint operator on $\mathcal{D}$ with respect to $I : \mathcal{C}^T \to \mathcal{D}$, and vice versa.*

(The reader is invited to check that the standard domain-theoretic situations arise by taking T as the lifting monad on a category of predomains.) In particular, Kleisli exponentials $X \Rightarrow Y$ fit in this scheme, where the algebra structure $\alpha_{X,Y} : T(X \Rightarrow Y) \to X \Rightarrow Y$ is given as the adjoint mate of

$$T(X \Rightarrow Y) \times X \xrightarrow{\theta} T((X \Rightarrow Y) \times X) \xrightarrow{T\mathsf{ev}} T^2Y \xrightarrow{\mu} TY$$

where $\mathsf{ev} : (X \Rightarrow Y) \times X \to TY$ is the counit of the adjunction. We note that $\eta \circ \alpha_{X,Y} : T(X \Rightarrow Y) \to T(X \Rightarrow Y)$ corresponds to an eta-expansion in the λ_c-calculus. That is, if a term $\Gamma \vdash M : X \to Y$ represents an arrow $f : A \to T(X \Rightarrow Y)$ in $\mathcal{C}$, then $\Gamma \vdash \lambda x^X.M\,x : X \to Y$ represents $\eta \circ \alpha_{X,Y} \circ f : A \to T(X \Rightarrow Y)$. This observation is frequently used in distilling the axioms of the stable uniform cbv fixpoint operators below.

5.1 Axiomatization in the λ_c-Calculus

Using the λ_c-calculus as an internal language of $\mathcal{C}_T$, the equation $f^* = f \circ f^*$ on $X \Rightarrow Y$ can be represented as

$$F^* \;=\; \lambda x^X.F\,F^*\,x \quad \text{where } F = \lambda f^{X \to Y}.\lambda x^X.F\,f\,x : (X \to Y) \to X \to Y$$

The side condition $F = \lambda f^{X \to Y}.\lambda x^X.F\,f\,x$ means that F corresponds to an arrow in $\mathcal{C}(X \Rightarrow Y, X \Rightarrow Y)$, not $\mathcal{C}_T(X \Rightarrow Y, X \Rightarrow Y)$. However, the operator

$(-)^* : \mathcal{C}(X \Rightarrow Y, X \Rightarrow Y) \to \mathcal{C}(1, X \Rightarrow Y)$ can be equivalently axiomatized by a slightly different operator

$$(-)^\ddagger : \mathcal{C}(X \Rightarrow Y, T(X \Rightarrow Y)) \to \mathcal{C}(1, X \Rightarrow Y)$$

subject to $f^\ddagger = \alpha_{X,Y} \circ f \circ f^\ddagger$, with an additional condition $f^\ddagger = (\eta \circ \alpha_{X,Y} \circ f)^\ddagger$. In fact, we can define such a $(-)^\ddagger$ as $(\alpha_{X,Y} \circ (-))^*$ and conversely $(-)^*$ by $(\eta \circ (-))^\ddagger$, and it is easy to see that these are in bijective correspondence. The condition $f^\ddagger = \alpha_{X,Y} \circ f \circ f^\ddagger$, equivalently $\eta \circ f^\ddagger = \eta \circ \alpha_{X,Y} \circ f \circ f^\ddagger$, is axiomatized in the λ_c-calculus as (by recalling that $\eta \circ \alpha_{X,Y} \circ (-)$ gives an eta-expansion)

$$F^\ddagger \;=\; \lambda x.F\, F^\ddagger\, x \text{ for any value } F : (X \to Y) \to X \to Y$$

which is precisely the cbv fixpoint axiom. The additional condition $f^\ddagger = (\eta \circ \alpha_{X,Y} \circ f)^\ddagger$ is axiomatized as

$$F^\ddagger = (\lambda f.\lambda x.F\, f\, x)^\ddagger \quad \text{where } F \text{ is a value}$$

This is no other than the stability axiom. We thus obtain the first two axioms of our stable uniform cbv fixpoint operators, which are precisely modelled by T-fixpoint operators.

5.2 Uniformity Axiom

Finally, we shall see how the uniformity condition on T-fixpoint operators can be represented in the λ_c-calculus. By Proposition 2, we define uniformity with respect to $I : \mathcal{C}^T \to \mathcal{D}$; that is, we regard $H \in \mathcal{C}(X \Rightarrow Y, X' \Rightarrow Y')$ as "strict" (or "rigid" in our terminology) if it is an algebra morphism from $(X \Rightarrow Y, \alpha_{X,Y})$ to $(X' \Rightarrow Y', \alpha_{X',Y'})$.[2] Spelling out this condition, we ask H to satisfy $H \circ \alpha_{X,Y} = \alpha_{X',Y'} \circ T(H)$, equivalently $T(H) \circ \eta \circ \alpha_{X,Y} = \eta \circ \alpha_{X',Y'} \circ T(H)$. In terms of the λ_c-calculus, this means that an eta-expansion commutes with the application of H; therefore, in the λ_c-calculus, we ask $H : (X \to Y) \to X' \to Y'$ to be a value such that

$$H(\lambda x^X.M\, x) \;=\; \lambda y^{X'}.H\, M\, y \;:\; X' \to Y'$$

holds for any $M : X \to Y$. We have called such an H rigid, and defined the uniformity condition with respect to such rigid functionals.

Theorem 1. *The computational models with a uniform T-fixpoint operator provide a sound and complete class of models of the computational lambda calculus with a stable uniform call-by-value fixpoint operator.*

[2] A characterisation of rigid functions (on computation types) in the same spirit is given in Filinski's thesis [5] (Section 2.2.2) though unrelated to the uniformity of fixpoint operators.

6 Recursion from Iteration Revisited

6.1 Iteration in the Category of Continuations

Let $\mathcal{C}$ be a response category with a response object R. An *iterator* on the category of continuations $R^{\mathcal{C}}$ is a family of functions $(-)_* : R^{\mathcal{C}}(A,A) \to R^{\mathcal{C}}(A,0)$ so that $f_* = f_* \circ f$ holds for $f \in R^{\mathcal{C}}(A,A)$. Spelling out this definition in $\mathcal{C}$, to give an iterator on $R^{\mathcal{C}}$ is to give a family of functions $(-)^* : \mathcal{C}(R^A, R^A) \to \mathcal{C}(1, R^A)$ so that $f^* = f \circ f^*$ holds for $f \in \mathcal{C}(R^A, R^A)$. Thus an iterator on $R^{\mathcal{C}}$ (hence in the cbv $\lambda\mu$-calculus) is no other than a fixpoint operator on $\mathcal{C}$ (hence the target call-by-name calculus) restricted on objects of the form R^A ("negative objects").

Example 1. We give a simple-minded model of the cbv $\lambda\mu$-calculus with an iterator. Let $\mathcal{C}$ be the category of ω-cpos (possibly without bottom) and continuous maps, and let R be an ω-cpo with bottom. Since $\mathcal{C}$ is a cartesian closed category with finite coproducts, it serves as a response category with the response object R. Moreover there is a least fixpoint operator on the negative objects R^A because R^A has a bottom element, thus we have an iterator on $R^{\mathcal{C}}$ (which in fact is a unique uniform iterator in the sense below).

6.2 Relation to Uniform T-Fixpoint Operators

For any object A, the negative object R^A canonically has a T-algebra structure $\alpha_A = \lambda m^{R^{R^{R^A}}}.\lambda x^A.m\,(\lambda f^{R^A}.f\,x) : R^{R^{R^A}} \to R^A$ for the monad $T = R^{R^{(-)}}$. Thus the consideration on the uniform T-fixpoint operators applies to this setting: if this computational model has a uniform T-fixpoint operator, then we have a fixpoint operator on negative objects, hence we can model an iterator of the cbv $\lambda\mu$-calculus in the category of continuations.

Conversely, if we have an iterator on $R^{\mathcal{C}}$, then it corresponds to a fixpoint operator on negative objects in $\mathcal{C}$, which of course include objects of the form $TA = R^{R^A}$. Therefore we obtain a T-fixpoint operator. It is then natural to expect that, if the iterator satisfies a suitable uniformity condition, then it bijectively corresponds to a uniform T-fixpoint operator. This uniformity condition on an iterator must be determined again with respect to the category of algebras $\mathcal{C}^T$. So we regard $h \in R^{\mathcal{C}}(A,B) \simeq \mathcal{C}(R^B, R^A)$ as "strict" ("total" in our terminology) when its counterpart in $\mathcal{C}(R^B, R^A)$ is an algebra morphism from (R^B, α_B) to (R^A, α_A), i.e., $h \circ \alpha_B = \alpha_A \circ R^{R^h}$ holds in $\mathcal{C}$. We say that an iterator $(-)_*$ on $R^{\mathcal{C}}$ is *uniform* if $f_* = g_* \circ h$ holds for $f : A \to A$, $g : B \to B$ and total $h : A \to B$ such that $h \circ f = g \circ h$.

Theorem 2. *Given a response category $\mathcal{C}$ with a response object R, to give a uniform $R^{R^{(-)}}$-fixpoint operator on $\mathcal{C}$ is to give a uniform iterator on $R^{\mathcal{C}}$.*

Fortunately, the condition to be an algebra morphism is naturally represented in a cbv $\lambda\mu$-theory. A value $h : A \to B$ represents an algebra morphism if and only if

$$x : A \vdash \mathsf{let}\ y^B\ \mathsf{be}\ h\,x\ \mathsf{in}\ \lambda k^{\neg B}.k\,y\ =\ \lambda k^{\neg B}.k(h\,x)\ :\ \neg\neg B$$

holds – in fact, the CPS translation of this equation is no other than the equation $h \circ \alpha_B = \alpha_A \circ R^{R^h}$. By Lemma 1, in a cbv $\lambda\mu$-theory, this requirement is equivalent to saying that $h\,x$ is a central term for each value x (this also implies Proposition 1). Therefore we obtain the uniformity condition for an iterator in Section 3.

Theorem 3. *In a cbv $\lambda\mu$-theory, there is a bijective correspondence between the stable uniform cbv fixpoint operators and the uniform iterators.*

6.3 On Filinski's Uniformity

In [4] Filinski introduced uniformity principles for both cbv fixpoint operators and iterators, for establishing a bijective correspondence between them. While his definitions turn out to be sufficient for his purpose, in retrospect they seem to be somewhat ad hoc and are strictly weaker than our uniformity principles. Here we give a brief comparison.

First, Filinski calls a value $h : \sigma \to \tau$ "total" when $h\,v$ is a *value* for each value $v : \sigma$. However, while a value is always central, the converse is not true. Note that, while the notion of centre is uniquely determined for each cbv $\lambda\mu$-theory (and category of continuations), the notion of value is not canonically determined (a category of continuations can arise from different response categories [17]). Since the uniformity principle is determined not in terms of the base category $\mathcal{C}$ but in terms of the category of algebras $\mathcal{C}^T$, it seems natural that it corresponds to the notion of centre which is determined not by $\mathcal{C}$ but by $\mathcal{C}_T$.

Second, Filinski calls a value $H : (\sigma \to \tau) \to \sigma' \to \tau'$ "rigid" when there are total $h_1 : \sigma' \to \tau \to \tau'$ and $h_2 : \sigma' \to \sigma$ such that

$$H = \lambda f^{\sigma\to\tau}.\lambda y^{\sigma'}.h_1\, y\, (f\, (h_2\, y)) : (\sigma \to \tau) \to \sigma' \to \tau'$$

holds. It is easily checked that if H is rigid in the sense of Filinski, it is also rigid in our sense – but the converse does not hold, even if we change the notion of total values to ours (for instance, $\mathsf{switch}_{\sigma,\tau}$ in Section 3 is not rigid in the sense of Filinski). By closely inspecting the correspondence of rigid functionals and total functions via the $\mathsf{step}/\mathsf{pets}$ and switch constructions, we can strengthen Filinski's formulation to match ours:

Proposition 3. *In a cbv $\lambda\mu$-theory, $H : (\sigma \to \tau) \to \sigma' \to \tau'$ is rigid if and only if there are total $h_1 : \sigma' \to \tau \to \tau'$ and $h_2 : (\sigma' \times \neg\tau') \to \sigma$ such that $H = \lambda f^{\sigma\to\tau}.\lambda y^{\sigma'}.\mu\gamma^{\tau'}.[\gamma](h_1\, y\, (f\, (h_2\, \langle y, \lambda z^{\tau'}.[\gamma]z\rangle)))$ holds.*

This subsumes Filinski's rigid functionals as special cases where h_2 does not use the second argument.

Remark 3. Filinski's uniformity principle in [4] takes the following form: if H is rigid and $H \circ (\lambda f.\lambda x.F\, f\, x) = G \circ H$ holds, then $H\,(\mathsf{fix}^{\mathsf{v}} F) = \mathsf{fix}^{\mathsf{v}}\, G$. It follows that this condition is equivalent to our stability and uniformity axioms.

7 Conclusion and Further Work

We have proposed an axiomatization of fixpoint operators in typed call-by-value programming languages, and have shown that it can be justified in two different ways: as a sound and complete axiomatization for uniform T-fixpoint operators of Simpson and Plotkin [18], and also by Filinski's bijective correspondence between recursion and iteration under the presence of first-class continuations [4]. We also have shown that these results are closely related, by inspecting the semantic structure behind Filinski's construction, which turns out to be a special case of the uniform T-fixpoint operators.

Towards practical principles for call-by-value recursion. We think that our axioms are reasonably simple, and we expect they can be a practical tool for reasoning about call-by-value programs involving recursion, just in the same way as the equational theory of the computational lambda calculus is the theoretical basis of the theory of A-normal forms [16,6].

It is an interesting challenge to strengthen the axioms in some systematic ways. For instance, by adding other natural axioms on an iterator under the presence of first-class controls, one may derive the corresponding axioms on the cbv fixpoint operator. In particular, we note that the *dinaturality* $\mathsf{loop}\,(g \circ f) = \mathsf{loop}\,(f \circ g) \circ f$ on an iterator loop precisely amounts to the axiom $\mathsf{fix}^{\mathsf{v}}\,(G \circ F) = \lambda x.G\,(\mathsf{fix}^{\mathsf{v}}\,(F \circ G))\,x$ on the corresponding cbv fixpoint operator $\mathsf{fix}^{\mathsf{v}}$. Similarly, the *diagonal property* on the iterator $\mathsf{loop}\,(\lambda x.\mu\alpha.[\alpha,\alpha](f\,x)) = \mathsf{loop}\,(\lambda x.\mu\alpha.\mathsf{loop}\,(\lambda y.\mu\beta.[\alpha,\beta](f\,y))\,x)$ corresponds to that on the fixpoint operator $\mathsf{fix}^{\mathsf{v}}\,(\lambda f.F\,f\,f) = \mathsf{fix}^{\mathsf{v}}\,(\lambda f.\mathsf{fix}^{\mathsf{v}}\,(\lambda g.F\,f\,g))$. These can be seen axiomatizing the call-by-value counterpart of the Conway theories [1,9]. One may further consider the call-by-value version of the Bekič property (another equivalent axiomatization of these properties [9]) along this line, which could be used for reasoning about mutual recursion.

Another promising direction is the approach based on *fixpoint objects* [2], as a uniform T-fixpoint operator is canonically derived from a fixpoint object whose universal property implies strong proof principles. For instance, in Example 1, a uniform iterator is unique because the monad $R^{R^{(-)}}$ has a fixpoint object. For the setting with first-class controls, it might be fruitful to study the implications of the existence of a fixpoint object of continuation monads.

Relating recursion in call-by-name and in call-by-value. The results reported here can be nicely combined with *Filinski's duality* [3] between call-by-value and call-by-name languages with first-class control primitives. In his MSc thesis [11], the second author demonstrates that recursion in the *call-by-name $\lambda\mu$-calculus* [13] exactly corresponds to iteration in the call-by-value $\lambda\mu$-calculus via this duality, by extending Selinger's work [17]. Together with the observation in this paper, we obtain a bijective correspondence between call-by-name recursion and call-by-value recursion, which seems to open a way to relate the reasoning principles on recursive computations under these two calling strategies.

Acknowledgements. We thank Shin-ya Katsumata for helpful discussions, and the anonymous reviewers for numerous suggestions.

References

1. Bloom, S. and Esik, Z. (1993) *Iteration Theories*. EATCS Monographs on Theoretical Computer Science, Springer-Verlag.
2. Crole, R.L. and Pitts, A.M. (1992) New foundations for fixpoint computations: FIX-hyperdoctrines and the FIX-logic. *Inform. and Comput.* **98**(2), 171–210.
3. Filinski, A. (1989) Declarative continuations: an investigation of duality in programming language semantics. In *Proc. Category Theory and Computer Science*, Springer Lecture Notes in Comput. Sci. **389**, pp. 224–249.
4. Filinski, A.(1994) Recursion from iteration. *Lisp and Symbolic Comput.***7**(1),11–38.
5. Filinski, A. (1996) *Controlling Effects*. PhD thesis, Carnegie Mellon University, CMU-CS-96-119.
6. Flanagan, C., Sabry, A., Duba, B.F. and Felleisen, M. (1993) The essence of compiling with continuations. In *Proc. ACM Conference on Programming Languages Design and Implementation*, pp. 237–247.
7. Führmann, C. (2000) *The Structure of Call-by-Value*. PhD thesis, University of Edinburgh.
8. Harper, R., Duba, B.F. and MacQueen, D. (1993) Typing first-class continuations in ML. *J. Funct. Programming* **3**(4), 465–484.
9. Hasegawa, M. (1997) *Models of Sharing Graphs: A Categorical Semantics of* let *and* letrec. PhD thesis, University of Edinburgh, ECS-LFCS-97-360; also in Distinguished Dissertation Series, Springer-Verlag, 1999.
10. Hofmann, M. (1995) Sound and complete axiomatisations of call-by-value control operators. *Math. Structures Comput. Sci.* **5**(4), 461–482.
11. Kakutani, Y. (2001) *Duality between Call-by-Name Recursion and Call-by-Value Iteration*. MSc thesis, Kyoto University.
12. Moggi, E. (1989) Computational lambda-calculus and monads. In *Proc. 4th Annual Symposium on Logic in Computer Science*, pp. 14–23.
13. Parigot, M. (1992) $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In *Proc. International Conference on Logic Programming and Automated Reasoning*, Springer Lecture Notes in Comput. Sci. **624**, pp. 190–201.
14. Plotkin, G.D. (1975) Call-by-name, call-by-value, and the λ-calculus. *Theoret. Comput. Sci.* **1**(1), 125–159.
15. Power, A.J. and Robinson, E.P. (1997) Premonoidal categories and notions of computation. *Math. Structures Comput. Sci.* **7**(5), 453–468.
16. Sabry, A. and Felleisen, M. (1992) Reasoning about programs in continuation-passing style. In *Proc. ACM Conference on Lisp and Functional Programming*, pp. 288–298; extended version in *Lisp and Symbolic Comput.* **6**(3/4), 289–360, 1993.
17. Selinger, P. (2001) Control categories and duality: on the categorical semantics of the lambda-mu calculus. To appear in *Math. Structures Comput. Sci.*
18. Simpson, A.K. and Plotkin, G.D. (2000) Complete axioms for categorical fixed-point operators. In *Proc. 15th Annual Symposium on Logic in Computer Science*.
19. Thielecke, H. (1997) *Categorical Structure of Continuation Passing Style*. PhD thesis, University of Edinburgh, ECS-LFCS-97-376.
20. Thielecke, H. (1999) Using a continuation twice and its implications for the expressive power of call/cc. *Higher-Order and Symbolic Comput.* **12**(1), 47–73.