A Syntactic View of Computational Adequacy

Marco Devesas Campos(✉) and Paul Blain Levy

School of Computer Science, University of Birmingham, Birmingham, UK
{m.devesascampos,pbl}@cs.bham.ac.uk

Abstract. When presenting a denotational semantics of a language with recursion, it is necessary to show that the semantics is computationally adequate, i.e. that every divergent term denotes the "bottom" element of a domain.

We explain how to view such a theorem as a purely syntactic result. Any theory (congruence) that includes basic laws and is closed under an infinitary rule that we call "rational continuity" has the property that every divergent term is equated with the divergent constant. Therefore, to prove a model adequate, it suffices to show that it validates the basic laws and the rational continuity rule. While this approach was inspired by the categorical, ordered framework of Abramsky et al., neither category theory nor order is needed.

The purpose of the paper is to present this syntactic result for call-by-push-value extended with term-level recursion and polymorphic types. Our account begins with PCF, then includes sum types, then moves to call-by-push-value, and finally includes polymorphic types.

1 Introduction

Models of Recursion. A conventional denotational account of a language with recursion proceeds as follows. First define the syntax and operational semantics. Then give a denotational model. Lastly, prove *soundness*, i.e. if t evaluates to u (written $t \Downarrow u$) then $[\![t]\!] = [\![u]\!]$, and *adequacy*, i.e. if t diverges (written $t \Uparrow$) then $[\![t]\!] = \bot$.

Because it is often convenient to structure a model categorically, Fiore and Plotkin (1994) gave categorical axioms on a model that imply (soundness and) adequacy. Crucially, in their work, as detailed by Fiore (1996), a model is required to be "ω**Cpo**-enriched", meaning that a term denotes an element of a pointed ω-cpo (poset with least element $\bot$ and suprema of all increasing ω-chains), and a term constructor is ω-continuous (preserves suprema of ω-chains). Thus (for a call-by-name language) a term $x : A \vdash t : A$ gives a continuous endofunction f, and the recursion $\mathbf{rec}\, x\,.\, M$ denotes the supremum of $(f^n \bot)_{n \in \mathbb{N}}$, the least (pre)fixpoint of f.

P. B. Levy—Research Supported by UK EPSRC Grant EP/N023757/1.

C. Baier and U. Dal Lago (Eds.): FOSSACS 2018, LNCS 10803, pp. 71–87, 2018.
https://doi.org/10.1007/978-3-319-89366-2_4

However, for the models of Abramsky et al. (2000), Abramsky and McCusker (1997), and McCusker (1998), the requirement of $\omega\mathbf{Cpo}$-enrichment is too restrictive, because the posets arising do not have suprema of *all* increasing ω-chains (Normann 2006). So these papers use a more relaxed ordered framework where the only suprema that must be preserved are those of chains $(f^n \bot)_{n\in\mathbb{N}}$ of iterated applications. This means that any so called *rational chain* $(g \circ f^n \bot)_{n\in\mathbb{N}}$ has an upper bound given by $g\,(\bigsqcup f^n \bot)$—a property known as *rational continuity* (Wright et al. 1976; cf. also Bloom and Ésik 1993).

Recursion but Rationally. Our goal is to give an even more relaxed version of this "rational" framework for adequacy; one that uses no category theory, order or denotational model. It could be viewed as a purely syntactic result: a property of a *theory* (congruence) $\approx$ rather than of a model. Thus we want $t \Downarrow u$ to imply $t \approx u$, and $t \Uparrow$ to imply $t \approx \Omega$, where Ω is a divergent constant. The benefit of such a result is to modularize the narrative described at the start; we can get adequacy out of the way before we start studying categorical and denotational semantics.

Rational Continuity. Currently we have accomplished this goal for term-level recursion and polymorphic types. (Recursive and existential types are left to future work; see Sect. 6). Our result is that any theory (congruence) $\approx$ will be sound and adequate provided it (a) contains the β-laws, fixpoint law and strictness laws and (b) is closed under an infinitary rule called *rational continuity.* This rule says (for a call-by-name language) that if $C[\mathbf{rec}^n\, x\,.\,t] \approx D[\mathbf{rec}^n\, x\,.\,t]$ for infinitely many $n \in \mathbb{N}$, then $C[\mathbf{rec}\, x\,.\,t] \approx D[\mathbf{rec}\, x\,.\,t]$. Here we write $\mathbf{rec}^n\, x\,.\,t$ for the nth *approximant to recursion*, defined by the clauses $\mathbf{rec}^0\, x\,.\,t := \Omega$ and $\mathbf{rec}^{n+1}\, x\,.\,t := t[\mathbf{rec}^n\, x\,.\,t/x]$.

Plan. To include both call-by-value (CBV) and call-by-name (CBN), we have established our result for call-by-push-value. The latter has both value types and computation types, but the treatment of value types in our proof is more complicated, so we begin in the CBN setting, which has only computation types. Our CBN account itself begins with PCF, which has only base types and function types; we then include sum types, using a proof method adapted from McCusker (1998). Next we move to call-by-push-value, and use *ultimate pattern matching* of values (Lassen and Levy 2008) to treat the value types. Finally we include polymorphic types.

Related Work. Adequacy of topos models has been studied using an internal language (Simpson 2004). Other adequacy results for polymorphic models include realizability semantics (Møgelberg 2009) and game semantics (Laird 2013).

2 PCF

Language. We begin by introducing a version of Plotkin's PCF (1997) that replaces fixpoint combinators with recursion operators and an explicit divergence construct Ω (Table 1). As per usual, terms are taken up to α-equivalence. The set

Table 1. PCF

Types $\quad T, U = \mathrm{Bool} \mid \mathrm{Nat} \mid T \to U$

Typing

$$\frac{(x : T \in \Gamma)}{\Gamma \vdash x : T} \qquad \frac{}{\Gamma \vdash tt : \mathrm{Bool}} \qquad \frac{}{\Gamma \vdash ff : \mathrm{Bool}} \qquad \frac{\Gamma \vdash t : \mathrm{Bool} \quad \Gamma \vdash u : T \quad \Gamma \vdash q : T}{\Gamma \vdash \mathbf{if}\, t\, \mathbf{then}\, u\, \mathbf{else}\, q : T}$$

$$\frac{}{\Gamma \vdash \mathbf{zero} : \mathrm{Nat}} \qquad \frac{\Gamma \vdash t : \mathrm{Nat}}{\Gamma \vdash \mathbf{succ}\, t : \mathrm{Nat}} \qquad \frac{\Gamma \vdash t : \mathrm{Nat}}{\Gamma \vdash \mathbf{pred}\, t : \mathrm{Nat}} \qquad \frac{\Gamma \vdash t : \mathrm{Nat}}{\Gamma \vdash \mathbf{iszero}\, t : \mathrm{Bool}}$$

$$\frac{x : T, \Gamma \vdash t : U}{\Gamma \vdash \lambda x.t : T \to U} \qquad \frac{\Gamma \vdash t : T \to U \quad \Gamma \vdash u : T}{\Gamma \vdash tu : U} \qquad \frac{}{\Gamma \vdash \Omega : T} \qquad \frac{x : T, \Gamma \vdash t : T}{\Gamma \vdash \mathbf{rec}\, x\, .\, t : T}$$

Reduction

$$\frac{}{tt \Downarrow tt} \qquad \frac{}{ff \Downarrow ff} \qquad \frac{t \Downarrow tt \quad u \Downarrow v}{\mathbf{if}\, t\, \mathbf{then}\, u\, \mathbf{else}\, q \Downarrow v} \qquad \frac{t \Downarrow ff \quad q \Downarrow v}{\mathbf{if}\, t\, \mathbf{then}\, u\, \mathbf{else}\, q \Downarrow v}$$

$$\frac{}{\mathbf{zero} \Downarrow \mathbf{zero}} \qquad \frac{t \Downarrow v}{\mathbf{succ}\, t \Downarrow \mathbf{succ}\, v} \qquad \frac{t \Downarrow \mathbf{succ}\, v}{\mathbf{pred}\, t \Downarrow v} \qquad \frac{t \Downarrow \mathbf{zero}}{\mathbf{iszero}\, t \Downarrow tt} \qquad \frac{t \Downarrow \mathbf{succ}\, v}{\mathbf{iszero}\, t \Downarrow ff}$$

$$\frac{}{\lambda x.t \Downarrow \lambda x.t} \qquad \frac{t \Downarrow (\lambda x.t') \quad t'[u/x] \Downarrow v}{tu \Downarrow v} \qquad \frac{t[\mathbf{rec}\, x\, .\, t/x] \Downarrow v}{\mathbf{rec}\, x\, .\, t \Downarrow v}$$

of closed terms of type T will be denoted by CTerms^T and that of normal forms by NF^T. For a closed term t there is at most one v such that $t \Downarrow v$; when there is none we say it diverges and represent this by $t \Uparrow$.

2.1 A Rationally Continuous Theory of PCF

The Theory. A *congruence on terms* is a type-indexed equivalence relation on closed terms of said type satisfying $t \approx t' \implies C[t] \approx C[t']$ for any context $C[-]$ where the hole is closed. (We omit type annotations.) A congruence is a *rationally continuous β-Ω-fix theory* if it also satisfies the rules in Table 2.

The basis for the theory are the obvious β rules that mimic the reduction rules. In a similar vein, the fixpoint rule establishes that each recursive term is the fixpoint of a substitution. These rules alone are enough to establish the soundness of the theory with respect to reduction.

Proposition 1 (Soundness). *Any congruence $\approx$ satisfying the β and fixpoint rules (Table 2) is sound: $t \Downarrow r \implies t \approx r$.*

A Converse. Our sights now turn to proving that divergent terms are identical to Ω. The extra requirement calls for a more refined theory that can more closely mirror the behaviour of reduction. The last two sets of equations in Table 2 fill the gaps in what the reduction rules *don't* say about divergence. The first

Table 2. Rationally continuous β-Ω-fix theory of PCF

Basis An equivalence relation $\approx$ on closed terms satisfying compatibility:

$$\text{for any (closed) } C[-],\ t \approx u \implies C[t] \approx C[u]$$

β Rules

$$\mathbf{if}\ tt\ \mathbf{then}\ u\ \mathbf{else}\ q \approx u \qquad \mathbf{if}\ ff\ \mathbf{then}\ u\ \mathbf{else}\ q \approx q \qquad \mathbf{iszero}\ \mathbf{zero} \approx tt$$
$$\mathbf{iszero}\ \mathbf{succ}^{n+1}\ \mathbf{zero} \approx ff \qquad \mathbf{pred}\ \mathbf{succ}^{n+1}\ \mathbf{zero} \approx \mathbf{succ}^{n}\ \mathbf{zero} \qquad (\lambda x.t)u \approx t[u/x]$$

Fixpoint Rule

$$\mathbf{rec}\, x\,.\, t \approx t[\mathbf{rec}\, x\,.\, t/x]$$

Divergence Rules

$$\Omega u \approx \Omega \qquad \mathbf{if}\ \Omega\ \mathbf{then}\ u\ \mathbf{else}\ q \approx \Omega \qquad \mathbf{iszero}\ \Omega \approx \Omega$$
$$\mathbf{succ}\ \Omega \approx \Omega \qquad \mathbf{pred}\ \Omega \approx \Omega \qquad \mathbf{pred}\ \mathbf{zero} \approx \Omega$$

Rational Continuity for $x : T \vdash t : T$

$$\frac{\exists^{\infty} n.C[\mathbf{rec}^n\, x\,.\, t] \approx D[\mathbf{rec}^n\, x\,.\, t]}{C[\mathbf{rec}\, x\,.\, t] \approx D[\mathbf{rec}\, x\,.\, t]}$$

where $\mathbf{rec}^0\, x\,.\, t = \Omega$ and $\mathbf{rec}^{n+1}\, x\,.\, t = t[\mathbf{rec}^n\, x\,.\, t/x]$.

relates to the strictness of the operators: divergence of an argument leads to the divergence of the operator, e.g., $\Omega u \approx \Omega$. The second is the rational continuity rule presented in the introduction.

Rational Continuity and Chains. To prove adequacy, one often has to re-write or equate certain terms built with recursion either with some constant or as the unrolling of the recursive term a few times. In cpo models, continuity and compositionality of the interpretations validate the following rule

$$\frac{\forall n \in \mathbb{N}.[\![C[\mathbf{rec}^n\, x\,.\, t]]\!] = [\![D[\mathbf{rec}^n\, x\,.\, t]]\!]}{[\![C[\mathbf{rec}\, x\,.\, t]]\!] = [\![D[\mathbf{rec}\, x\,.\, t]]\!]}$$

But this can be further weakened by requiring only equality *at infinitely many* n, for then one would still be able to define chains with exactly the same least upper bounds. We write $\exists^{\infty} n.P(n)$ to mean *there exist infinitely many n in $\mathbb{N}$ for which $P(n)$ holds.* This leads us to the syntactic continuity rule in Table 2. Since adequacy refers solely to closed terms, we only require this property for $x : T \vdash t : T$—and therefore $\mathbf{rec}^n\, x\,.\, t$ and $\mathbf{rec}\, x\,.\, t$ are closed. Similarly, by a *rational chain* we mean a chain of the form $C[\mathbf{rec}^n x\,.\, t]$ for infinitely many $n \in \mathbb{N}$, and by its limit we mean the term $C[\mathbf{rec}\, x\,.\, t]$.

2.2 Adequacy

The Claim. We now embark on the syntactic journey towards a proof we have an adequate theory—formally, that $t \Uparrow \implies t \approx \Omega$. By the aforementioned reasons the proof follows the usual approaches by replacing closure under bottom elements and least upper bounds of the relevant chains with closure under divergence and limits of rational chains.

Approximations. First we define abstractly[1] the notion of an approximation candidate between terms and the values they approximate; these are then extended to relations on terms. The concrete relations we use for each type are given by certain actions on approximation candidates (cf., e.g., Pitts 2000). When using the result of an action ϕ on approximation candidates $\lhd_1, \ldots, \lhd_n$ infix, we will sometimes surround the result with brackets, as in $t \ \langle \phi(\lhd_1, \ldots, \lhd_n) \rangle \ u$, to aid readability.

Definition 1 (Approximation Candidates). *An approximation candidate $\lhd$ for a type T is a subset of $CTerms^T \times NFs^T$ s.t.:*

1. *$\approx$ Extension: $t \approx t'$ and $t' \lhd v \implies t \lhd v$*
2. *Rational Admissibility: for $x : T \vdash t : T$*

$$(\exists^\infty n. C[\mathbf{rec}^n \, x \, . \, t] \lhd v) \implies C[\mathbf{rec} \, x \, . \, t] \lhd v$$

Proposition 2. *If $\lhd$ is an approximation candidate for type T, then the binary relation on $CTerms^T$ defined by*

$$t \lhd^c u \iff t \approx \Omega \text{ or } (\exists v. u \Downarrow v \text{ and } t \lhd v)$$

satisfies the following properties:

1. *Ω Property: $\Omega \lhd^c u$, for any $u \in CTerms^T$*
2. *$\approx$ Extension: $t \approx t'$ and $t' \lhd^c u \implies t \lhd^c u$*
3. *$\Downarrow$ Extension: $t \lhd^c u$ and $(\forall v. u \Downarrow v \implies u' \Downarrow v) \implies t \lhd^c u'$*
4. *Rational Admissibility: for $x : T \vdash t : T$*

$$(\exists^\infty n. C[\mathbf{rec}^n \, x \ . \ t] \lhd^c u) \implies C[\mathbf{rec} \, x \, . \, t] \lhd^c u$$

Proof. To give a taste of how the proofs go using rational admissibility, assume we have $\exists^\infty n. C[\mathbf{rec}^n \, x \, . \, t] \lhd^c u$. From the definition, one of two options (possibly both) is true: that an infinite number of terms on the left are identical to Ω; or that for an infinite series of m, $C[\mathbf{rec}^m \, x \, . \, t]$ is related to the value v that u reduces to (determinism of reduction is paramount here). Admissibility then follows by rational continuity in the first case (using the obvious constant context), and by admissibility of $\lhd$ (Definition 1) in the second.

[1] Anticipating our treatment of polymorphism in Sect. 4, we have purposefully set up here a proof structure in the style of Girard (1989).

Proposition 3 (Base Type Actions). *The two binary relations* $\lhd_{Bool} \subseteq CTerms^{Bool} \times NFs^{Bool}$ *and* $\lhd_{Nat} \subseteq CTerms^{Nat} \times NFs^{Nat}$ *defined by*

$$t \lhd_{Bool} v \iff t \approx v \quad \text{and} \quad t \lhd_{Nat} v \iff t \approx v$$

are approximation candidates for Bool and Nat.

Proposition 4 (Arrow Action). *Given approximation candidates* $\lhd_T$ *for* T *and* $\lhd_U$ *for* U*, the binary relation between* $CTerms^{T\to U}$ *and* $NFs^{T\to U}$

$$t \langle \lhd_T \to \lhd_U \rangle \lambda x.u \iff \forall p \lhd_T^c q \;.\; tp \lhd_U^c u[q/x])$$

is an approximation candidate for $T \to U$*.*

Definition 2 (Approximation Relation). *The approximation relation* $\lhd_T$ *is the type-indexed family of approximation candidates defined by induction on types, where base types are covered by their respective actions (Proposition 3), and* $\lhd_{T\to U} = \lhd_T \to \lhd_U$ *(Proposition 4).*

Definition 3 (Environments). *Given a typing context* Γ*, an* environment σ *for* Γ *is a substitution that maps each* $x : T \in \Gamma$ *to a closed term of type* $\vdash \sigma(x) : T$*. If* σ_1 *and* σ_2 *are two such, we write* $\sigma_1 \lhd_\Gamma^c \sigma_2$ *to mean* $\sigma_1(x) \lhd_T^c \sigma_2(x)$ *for all* $x : T \in \Gamma$*.*

Proposition 5. *For any* $\Gamma \vdash t : T$ *and environments* $\sigma_1 \lhd_\Gamma^c \sigma_2$*,* $t[\sigma_1] \lhd_T^c t[\sigma_2]$*.*

Corollary 1 (Adequacy). *For every closed* $\vdash t : T$*,* $t \Uparrow \implies t \approx \Omega$*.*

Proof. Applying Proposition 5 to $\vdash t : T$ *(for the empty substitution), we conclude that* $t \lhd_T^c t$*; the definition of* $(-)^c$ *(Proposition 2) asserts, then, that either* $t \approx \Omega$ *or* $(t \Downarrow v$ and $t \lhd_T v)$*; whereby if* $t \Uparrow$*, it can only be that* $t \approx \Omega$*.*

3 PCF with Sums

The Extension. Sums provide a slight complication—but one which shows the adaptability of the method. The extension to call-by-name sums is presented in Table 3. With the new reduction rules come new β rules and divergence rules in the theory (Table 4). As before, reduction is deterministic and the theory is sound.

3.1 Adequacy

Action. The action for sums must reflect the structure of its parameters. That is for $\lhd_T$ we expect $t \lhd_{T+U} \mathbf{inl}\, u$ exactly when (modulo the theory) t decomposes into some $\mathbf{inl}\, t'$ for which $t' \lhd_T u$. The assertion of that existence, though, causes us a small hiccup[2] in proving that $- \lhd_{T+U} v$ is rationally admissible: If we have

[2] A hiccup that will be much amplified in the proof of admissibility for $\lhd_{FA}$ (Sect. 4).

Table 3. Extension of PCF with binary sums

Types

$$T, U = \ldots \mid T + U$$

Typing

$$\frac{\Gamma \vdash t : T}{\Gamma \vdash \mathbf{inl}\, t : T + U} \qquad \frac{\Gamma \vdash t : U}{\Gamma \vdash \mathbf{inr}\, t : T + U}$$

$$\frac{\Gamma \vdash t : T + T' \quad x : T, \Gamma \vdash u : U \quad y : T', \Gamma \vdash q : U}{\Gamma \vdash \mathbf{match}\, t\, \mathbf{as}\, \{\mathbf{inl}\, x.u\, ,\, \mathbf{inr}\, y.q\} : U}$$

Reduction

$$\frac{}{\mathbf{inl}\, t \Downarrow \mathbf{inl}\, t} \qquad \frac{}{\mathbf{inr}\, t \Downarrow \mathbf{inr}\, t} \qquad \frac{t \Downarrow \mathbf{inl}\, t' \quad u[t'/x] \Downarrow v}{\mathbf{match}\, t\, \mathbf{as}\, \{\mathbf{inl}\, x.u\, ,\, \mathbf{inr}\, y.q\} \Downarrow v}$$

$$\frac{t \Downarrow \mathbf{inr}\, t' \quad q[t'/x] \Downarrow v}{\mathbf{match}\, t\, \mathbf{as}\, \{\mathbf{inl}\, x.u\, ,\, \mathbf{inr}\, y.q\} \Downarrow v}$$

Table 4. Extension of the theory in Table 2 with binary sums

β Rules

$$\mathbf{match}\ \mathbf{inl}\, t\, \mathbf{as}\, \{\mathbf{inl}\, x.u\, ,\, \mathbf{inr}\, y.q\} \approx u[t/x] \qquad \mathbf{match}\ \mathbf{inr}\, t\, \mathbf{as}\, \{\mathbf{inl}\, x.u\, ,\, \mathbf{inr}\, y.q\} \approx q[t/x]$$

Divergence Rules

$$\mathbf{match}\, \Omega\, \mathbf{as}\, \{\mathbf{inl}\, x.u\, ,\, \mathbf{inr}\, y.q\} \approx \Omega$$

a series of $C[\mathbf{rec}^n\, x\, .\, t] \lhd_{T+U} \mathbf{inl}\, u$, then we know that each of the terms on the left must be identical to some $\mathbf{inl}\, t_n$ with $t_n \lhd_T u$—but do the t_n form a rational chain? It turns out that for every t, simply from the existence of $t \approx \mathbf{inl}\, t'$, and because each type is inhabited by Ω, there is a context that can extract directly the t' (up to equivalence, obviously) from the original term. (An idea we borrowed from McCusker 1998)

Lemma 1. *The contexts*

$$\mathcal{T}^l[-] = \mathbf{match}\ -\ \mathbf{as}\, \{\mathbf{inl}\, x.x\, ,\, \mathbf{inr}\, y.\Omega\}$$

$$\mathcal{T}^r[-] = \mathbf{match}\ -\ \mathbf{as}\, \{\mathbf{inl}\, x.\Omega\, ,\, \mathbf{inr}\, y.y\}$$

satisfy $t \approx \mathbf{inl}\, u \implies \mathcal{T}^l[t] \approx u$ *and* $t \approx \mathbf{inr}\, u \implies \mathcal{T}^r[t] \approx u$.

Proposition 6 (Sum Action). *Given approximation candidates* $\lhd_T$ *for* T *and* $\lhd_U$ *for* U, *the relation between* $CTerms^{T+U}$ *and* NFs^{T+U} *defined by*

$$t \left\langle \lhd_T + \lhd_U \right\rangle \mathbf{inl}\, v \iff (\exists t' \lhd_T^c u.t \approx \mathbf{inl}\, t')$$

$$t \, \langle \lhd_T + \lhd_U \rangle \, \mathbf{inr}\, v \iff (\exists t' \lhd_U^c u.t \approx \mathbf{inr}\, t')$$

is an approximation candidate for $A + B$.

Proof. For rational admissibility, the pre-condition must hold for (at least) one of the two clauses in the definition. Say we have $\exists^\infty n.C[\mathbf{rec}^n\, x\,.\,t] \; \langle \lhd_T + \lhd_U \rangle \; \mathbf{inl}\, u$ *with each term on the left equivalent to some* $\mathbf{inl}\, t_n$*; rewriting* $t_n \approx \mathcal{T}^l[C[\mathbf{rec}^n\, x\,.\,t]]$ *(Lemma 1) it follows that (Proposition 2)*

$$C[\mathbf{rec}^n\, x\,.\,t] \approx \mathbf{inl}\, \mathcal{T}^l[C[\mathbf{rec}^n\, x\,.\,t]] \text{ and } \mathcal{T}^l[C[\mathbf{rec}^n\, x\,.\,t]] \lhd_T^c u$$

An application of rational continuity of the theory, and one of rational admissibility of $\lhd_T^c$ *(again, Proposition 2) yields* $C[\mathbf{rec}\, x\,.\,t] \approx \mathbf{inl}\, \mathcal{T}^l[C[\mathbf{rec}\, x\,.\,t]]$ *and also* $\mathcal{T}^l[C[\mathbf{rec}\, x\,.\,t]] \lhd_T^c u$ *so that* $C[\mathbf{rec}\, x\,.\,t] \; \langle \lhd_T + \lhd_U \rangle \; \mathbf{inl}\, u$*. (Likewise for the right injection.)*

Adequacy. The rest of the proof of adequacy follows exactly as before. Approximation candidates for sums are derived by induction using the sum action; and with them we can extend Proposition 5.

4 Call-by-Push-Value

Values vs. Computations. We now turn to Call-by-push-value (Levy 2004). This language (Table 5) distinguishes between values and computations, with value types represented by A, A', etc., and computation types by $\underline{B}$, $\underline{B'}$, etc. The set of closed values of type A will be represented by Vals^A; that of closed computations by $\mathrm{Comps}^{\underline{B}}$. Variables always have value type. Here we include value products and sums, products of computation types $\underline{B} \mathbin{\pi} \underline{B'}$, types FA for computations aiming to return a value, and functions which in CBPV are computations taking values to computations. Central to CBPV, we also include value types $U\underline{B}$ of suspended computations of type $\underline{B}$—which can be of one of two forms.

Recursion. In addition to the usual **thunk**s of computations, we also have recursively defined thunks $\mathbf{threc}\, x.t$. An alternative would be to use recursive computations $\Gamma \vdash^c \mathbf{rec}\, x.t : \underline{B}$. Although the two are equivalent via the definitions $\mathbf{rec}\, x.t := \mathbf{force}\, \mathbf{threc}\, x.t$ and $\mathbf{threc}\, x.t := \mathbf{thunk}\, \mathbf{rec}\, x.t$, there are two reasons for preferring **threc**: One is that, in some denotational models (e.g. state or continuation passing), **threc** has a simpler denotation than **rec**. The other is that a treatment based on **threc** would be more easily adapted to call-by-value, where recursion and lambda are combined.

Evaluation. Evaluation (Table 6) pertains only to computations. To those on the co-domain side of the evaluation relation $\Downarrow$, we call the *terminal* computations or, alternatively, the normal forms; and their (typed-indexed) set is represented by $\mathrm{NFs}^{\underline{B}}$. Since we have two forms of thunked computations, the action of forcing one such into execution much act accordingly; this *unthunk*ing (a derived operation on the syntax) returns the computations suspended inside **thunk**s, or plucks out the computation from a $\mathbf{threc}\, x\,.\,t$ suitably instantiated by the recursive thunk itself—i. e. $t[\mathbf{threc}\, x\,.\,t/x]$. Note that reduction is deterministic.

Table 5. Call-by-push with recursion-value—syntax

Types

$$A, A', \ldots = 1 \mid A \times A' \mid 0 \mid A + A' \mid U\underline{B} \qquad \underline{B}, \underline{B'}, \ldots = FA \mid A \to \underline{B} \mid 1_{\Pi} \mid \underline{B} \,\Pi\, \underline{B'}$$

Typing

$$\frac{(x : A \in \Gamma)}{\Gamma \vdash^v x : A} \qquad \frac{\Gamma \vdash^v v : A \quad x : A, \Gamma \vdash^c t : \underline{B}}{\Gamma \vdash^c \mathbf{let}\, v\, \mathbf{be}\, x\,.\, t : \underline{B}} \qquad \frac{\Gamma \vdash^v v : A}{\Gamma \vdash^c \mathbf{return}\ v : FA}$$

$$\frac{\Gamma \vdash^c u : FA \quad x : A, \Gamma \vdash^c t : \underline{B}}{\Gamma \vdash^c u\, \mathbf{to}\, x.\, t : \underline{B}} \qquad \frac{\Gamma \vdash^c t : \underline{B}}{\Gamma \vdash^v \mathbf{thunk}\, t : U\underline{B}} \qquad \frac{\Gamma \vdash^v v : U\underline{B}}{\Gamma \vdash^c \mathbf{force}\, v : \underline{B}}$$

$$\frac{x : U\underline{B}, \Gamma \vdash^c t : \underline{B}}{\Gamma \vdash^v \mathbf{threc}\, x\,.\, t : U\underline{B}} \qquad \frac{}{\Gamma \vdash^c \Omega : \underline{B}} \qquad \frac{}{\Gamma \vdash^v \langle\rangle : 1}$$

$$\frac{\Gamma \vdash^v v : A \quad \Gamma \vdash^v v' : A'}{\Gamma \vdash^v \langle v, v'\rangle : A \times A'} \qquad \frac{\Gamma \vdash^v v : A \times A' \quad x : A, y : A', \Gamma \vdash^c t : \underline{B}}{\Gamma \vdash^c \mathbf{match}\, v\, \mathbf{as}\, \langle x, y\rangle\,.\, t : \underline{B}}$$

$$\frac{\Gamma \vdash^v v : 1 \quad \Gamma \vdash^c t : \underline{B}}{\Gamma \vdash^c \mathbf{match}\, v\, \mathbf{as}\, \langle\rangle\,.\, t : \underline{B}} \qquad \frac{\Gamma \vdash^v v : 0}{\Gamma \vdash^c \mathbf{match}\, v\, \mathbf{as}\, \{\} : \underline{B}} \qquad \frac{}{\Gamma \vdash^c \lambda\,\{\} : 1_{\Pi}}$$

$$\frac{\Gamma \vdash^c t : \underline{B} \quad \Gamma \vdash^c t' : \underline{B'}}{\Gamma \vdash^c \lambda\left\{{}^l.t, {}^r.t'\right\} : \underline{B} \,\Pi\, \underline{B'}} \qquad \frac{\Gamma \vdash^c t : \underline{B} \,\Pi\, \underline{B'}}{\Gamma \vdash^c t^l : \underline{B}} \qquad \frac{\Gamma \vdash^c t : \underline{B} \,\Pi\, \underline{B'}}{\Gamma \vdash^c t^r : \underline{B'}}$$

$$\frac{x : A, \Gamma \vdash^c t : \underline{B}}{\Gamma \vdash^c \lambda x.t : A \to \underline{B}} \qquad \frac{\Gamma \vdash^v v : A \quad \Gamma \vdash^c t : A \to \underline{B}}{\Gamma \vdash^c tv : \underline{B}} \qquad \frac{\Gamma \vdash^v v : A}{\Gamma \vdash^v \mathbf{inl}\, v : A + A'}$$

$$\frac{\Gamma \vdash^v v : A'}{\Gamma \vdash^v \mathbf{inr}\, v : A + A'} \qquad \frac{\Gamma \vdash^v v : A + A' \quad x : A, \Gamma \vdash^c t : \underline{B} \quad y : A', \Gamma \vdash^c t' : \underline{B}}{\Gamma \vdash^c \mathbf{match}\, v\, \mathbf{as}\, \{\mathbf{inl}\, x.t, \mathbf{inr}\, y.t'\} : \underline{B}}$$

Table 6. Call-by-push-value with recursion—reduction

$$\frac{}{\mathbf{return}\ v \Downarrow \mathbf{return}\ v} \qquad \frac{}{\lambda x.t \Downarrow \lambda x.t} \qquad \frac{}{\lambda\,\{\} \Downarrow \lambda\,\{\}} \qquad \frac{}{\lambda\left\{{}^l.t, {}^r.t'\right\} \Downarrow \lambda\left\{{}^l.t, {}^r.t'\right\}}$$

$$\frac{t[v/x] \Downarrow r}{\mathbf{let}\, v\, \mathbf{be}\, x\,.\, t \Downarrow r} \qquad \frac{t \Downarrow r}{\mathbf{match}\ \langle\rangle\ \mathbf{as}\ \langle\rangle\,.\, t \Downarrow r} \qquad \frac{t[v/x, v'/y] \Downarrow r}{\mathbf{match}\ \langle v, v'\rangle\ \mathbf{as}\ \langle x, y\rangle\,.\, t \Downarrow r}$$

$$\frac{t[v/x] \Downarrow r}{\mathbf{match}\ \mathbf{inl}\, v\, \mathbf{as}\, \{\mathbf{inl}\, x.t, \mathbf{inr}\, y.t'\} \Downarrow r} \qquad \frac{t'[v/y] \Downarrow r}{\mathbf{match}\ \mathbf{inr}\, v\, \mathbf{as}\, \{\mathbf{inl}\, x.t, \mathbf{inr}\, y.t'\} \Downarrow r}$$

$$\frac{t \Downarrow \lambda\left\{{}^l.u, {}^r.q\right\} \quad u \Downarrow r}{t^l \Downarrow r} \qquad \frac{t \Downarrow \lambda\left\{{}^l.u, {}^r.q\right\} \quad q \Downarrow r}{t^r \Downarrow r}$$

$$\frac{u \Downarrow \mathbf{return}\ v \quad t[v/x] \Downarrow r}{u\, \mathbf{to}\, x.\, t \Downarrow r} \qquad \frac{t \Downarrow \lambda x.u \quad u[v/x] \Downarrow r}{tv \Downarrow r} \qquad \frac{unthunk\, v \Downarrow r}{\mathbf{force}\, v \Downarrow r}$$

where $unthunk(\mathbf{thunk}\, t) = t$ and $unthunk(\mathbf{threc}\, x\,.\, t) = t[\mathbf{threc}\, x\,.\, t/x]$,

4.1 Theory

Theory. By a (CBPV) congruence on closed terms we mean a type-indexed equivalence relation $\approx$ on closed values and computations such that for all closed terms $t \approx t'$ and (value or computation) context $C[-]$ we have $C[t] \approx C[t']$, respectively. A congruence is a *rationally continuous β-Ω-fix theory* when it satisfies the rules in Table 7. Rational chains are now those built by the application of a context $C[-]$ to the (thunked) approximants $\mathbf{threc}^n$ of recursive thunks and which are defined by the clauses $\mathbf{threc}^0\, x\,.\,t = \mathbf{thunk}\,\Omega$ and $\mathbf{threc}^{n+1}\, x\,.\,t = \mathbf{thunk}\, t[\mathbf{threc}^n\, x\,.\,t/x]$; continuity is defined accordingly. Any congruence including the β and fixpoint rules is easily seen to be sound. We shall show that with the remaining rules it is also adequate.

Table 7. Call-by-push-value with recursion—rationally continuous β-Ω-fix theory

Basis An equivalence relation $\approx$ on closed terms satisfying compatibility:

$$\text{for any } C[-],\ t \approx u \implies C[t] \approx C[u] \qquad \text{for any } C[-],\ v \approx w \implies C[v] \approx C[w]$$

β Rules

$$\mathbf{let}\, v\, \mathbf{be}\, x\,.\,t \approx t[v/x] \qquad \mathbf{match}\ \langle\rangle\ \mathbf{as}\ \langle\rangle\,.\,t \approx t$$
$$\mathbf{match}\ \langle v, v'\rangle\ \mathbf{as}\ \langle x, y\rangle\,.\,t \approx t[v/x, v'/y]$$
$$\mathbf{match}\ \mathbf{inl}\, v\ \mathbf{as}\ \{\mathbf{inl}\, x.t, \mathbf{inr}\, y.t'\} \approx t[v/x]$$
$$\mathbf{match}\ \mathbf{inr}\, v\ \mathbf{as}\ \{\mathbf{inl}\, x.t, \mathbf{inr}\, y.t'\} \approx t'[v/y]$$
$$\left(\lambda\left\{{}^{l}.t, {}^{r}.t'\right\}\right)^{l} \approx t \qquad \left(\lambda\left\{{}^{l}.t, {}^{r}.t'\right\}\right)^{r} \approx t' \qquad \mathbf{return}\ v\ \mathbf{to}\, x.\, t \approx t[v/x]$$
$$\mathbf{force}\,\mathbf{thunk}\, t \approx t \qquad (\lambda x.t)v \approx t[v/x]$$

Fixpoint Rule $\qquad \mathbf{threc}\, x\,.\,t \approx \mathbf{thunk}\, t[\mathbf{threc}\, x\,.\,t/x]$

Divergence Rules $\qquad \Omega\ \mathbf{to}\, x.\, t \approx \Omega \qquad (\Omega)^l \approx \Omega \qquad (\Omega)^r \approx \Omega \qquad \Omega v \approx \Omega$

Rational Continuity for $x : U\underline{B} \vdash^c t : \underline{B}$ and $C[-]$, $D[-]$ computation contexts

$$\frac{\exists^{\infty} n. C[\mathbf{threc}^n\, x\,.\,t] \approx D[\mathbf{threc}^n\, x\,.\,t]}{C[\mathbf{threc}\, x\,.\,t] \approx D[\mathbf{threc}\, x\,.\,t]}$$

where $\mathbf{threc}^0\, x\,.\,t = \mathbf{thunk}\,\Omega$ and $\mathbf{threc}^{n+1}\, x\,.\,t = \mathbf{thunk}\, t[\mathbf{threc}^n\, x\,.\,t/x]$.

4.2 Adequacy

Values: Empty Shells. In the proof of adequacy for PCF with sums we were required to introduce the tests so that we could, metaphorically, peek inside the injections and transform the rational chains there into equivalent ones with the

properties we needed (cf. proof of Proposition 6). Here the problem expands to *all value types*. When checking rational admissibility, we need to decompose a value into its *ultimate pattern* and its constituent thunks (Lassen and Levy 2008, following ideas from Abramsky and McCusker 1997; also discernible in the work of Zeilberger 2008) and use those to find equivalent chains that can be used to establish adequacy.

Definition 4 (Ultimate Patterns). *The set of of ultimate patterns UP^A for a value type A is given by induction on the following rules: $-_{U\underline{B}} \in UP^{U\underline{B}}$, $\langle\rangle \in UP^1$ and*

$$\frac{p \in UP^A \quad p' \in UP^{A'}}{\langle p, p' \rangle \in UP^{A \times A'}} \qquad \frac{p \in UP^A}{\mathbf{inl}\, p \in UP^{A+A'}} \qquad \frac{p \in UP^{A'}}{\mathbf{inr}\, p \in UP^{A+A'}}$$

For a given ultimate pattern $p \in UP^A$ the finite sequence of hole-types in pattern p is given by induction by

$$H(-_{U\underline{B}}) = (U\underline{B}) \qquad H(\langle\rangle) = \epsilon \qquad H(\langle p, p' \rangle) = H(p) +\!\!+ H(p')$$
$$H(\mathbf{inl}\, p) = H(p) \qquad H(\mathbf{inr}\, p) = H(p)$$

Proposition 7 (Value Decomposition). *Given $\vdash^v v : A$, there is a unique $p \in UP^A$ and a unique sequence $(\vdash^v v_i : H(p)_i)_{i<|H(p)|}$—the filling—for which $v = p \mathbin{@} (v_i)_{i<|H(p)|}$, using the reassembly function*

$$(-_{U\underline{B}}) \mathbin{@} (v) = v \qquad \langle\rangle \mathbin{@} \epsilon = \langle\rangle$$
$$\mathbf{inl}\, p \mathbin{@} (v_i)_{i<|H(p)|} = \mathbf{inl}(p \mathbin{@} (v_i)_{i<|H(p)|})$$
$$\mathbf{inr}\, p \mathbin{@} (v_i)_{i<|H(p)|} = \mathbf{inr}(p \mathbin{@} (v_i)_{i<|H(p)|})$$
$$\langle p, p' \rangle \mathbin{@} \left((v_i)_{i<|H(p)|} +\!\!+ (v'_i)_{i<|H(p')|}\right) = \left\langle (p \mathbin{@} (v_i)_{i<|H(p)|}), (p' \mathbin{@} (v'_i)_{i<|H(p')|}) \right\rangle$$

Tests. Ultimate patterns let us define the tests that extract the computations embedded in a given value. Like in the PCF sum case, we can use them to define values that are equivalent to a given one but make use only of the latter. If the values are derived from some family of contexts for the holes, then we can derive an equivalent context from the respective ultimate pattern.

Definition 5. *For $p \in UP^A$, and $i < |H(p)|$, we define a context $\mathcal{T}_i^p[-]$ by induction on $p \in UP^A$ using the rules below. Note that when $\Gamma \vdash^v - : A$ the test has type $\Gamma \vdash^c \mathcal{T}_i^p[-] : \underline{B_i}$ where $U\underline{B_i} = H(p)_i$.*

$$\mathcal{T}_0^{-_{UB}}[-] = \mathbf{force}\, -$$
$$\mathcal{T}_i^{\mathbf{inl}\, p}[-] = \mathbf{match}\ -\ \mathbf{as}\ \{\mathbf{inl}\, x.\, \mathcal{T}_i^p[x],\ \mathbf{inr}\, y.\Omega\}$$
$$\mathcal{T}_i^{\mathbf{inr}\, p}[-] = \mathbf{match}\ -\ \mathbf{as}\ \{\mathbf{inl}\, x.\Omega,\ \mathbf{inr}\, y.\, \mathcal{T}_i^p[y]\}$$
$$\mathcal{T}_{i<|H(p)|}^{\langle p,p' \rangle}[-] = \mathbf{match}\, -\, \mathbf{as} < x, y > .\, \mathcal{T}_i^p[x]$$
$$\mathcal{T}_{i=|H(p)|+i'}^{\langle p,p' \rangle}[-] = \mathbf{match}\, -\, \mathbf{as} < x, y > .\, \mathcal{T}_{i'}^{p'}[y]$$

Proposition 8 (Tests Decompose). *Given a pattern $p \in UP^A$, a sequence $(\vdash w_i : H(p)_i)_{i<|H(p)|}$, and $i < |H(p)|$, we have $\mathcal{T}_i^p[p \mathbin{@} (w_i)_{i<|H(p)|}] \approx \mathbf{force}\, w_i$.*

Proposition 9. *For $\vdash^c t : FA$, and $p \in UP^A$, if $t \approx \mathbf{return}\, p \mathbin{@} (v_i)_{i<|H(p)|}$ then, successively:*

1. $\forall i < |H(p)|.\, \mathbf{thunk}(t\ \mathbf{to}\ x.\ \mathcal{T}_i^p[x]) \approx v_i$
2. $p \mathbin{@} (v_i)_{i<|H(p)|} \approx p \mathbin{@} (\mathbf{thunk}(t\ \mathbf{to}\ x.\ \mathcal{T}_i^p[x]))_{i<|H(p)|}$
3. $t \approx \mathbf{return}\, p \mathbin{@} (\mathbf{thunk}(t\ \mathbf{to}\ x.\ \mathcal{T}_i^p[x]))_{i<|H(p)|}$

Approximation Candidates. Unlike PCF where we have computations and normal forms, CBPV has three levels of syntax: values, terminals, and computations. For the purposes of defining the needed approximation candidates, terminals (read: normal forms) and computations, behave like their PCF counterparts and have (now) familiar definitions of approximation candidates. Approximation candidates for value types enforce that: only structurally similar values are related; that they are (left) closed under equivalence of their holes; and that they are closed under the usual chains.

Definition 6 (Approximation Candidates). *Given a value type A, an approximation candidate $\lhd$ for A is a subset of $Vals^A \times Vals^A$ such that*

1. *Structural Matching: $p \mathbin{@} (v_i)_i \lhd p' \mathbin{@} (w_i)_i \implies p = p'$*
2. *Computational $\approx$ Extension: if $p \mathbin{@} (v'_i)_{i<|H(p)|} \lhd p \mathbin{@} (w_i)_{i<|H(p)|}$ then*

$$(\forall i < |H(p)|.v_i \approx v'_i) \implies p \mathbin{@} (v_i)_{i<|H(p)|} \lhd p \mathbin{@} (w_i)_{i<|H(p)|}$$

3. *Rational Admissibility: for $x : U\underline{B} \vdash^c t : \underline{B}$*

$$(\exists^\infty n.V[\mathbf{threc}^n x.t] \lhd w) \implies V[\mathbf{threc}\, x\,.\, t] \lhd w$$

Given a computation type $\underline{B}$, an approximation candidate $\lhd$ for $\underline{B}$ is a subset of $Comps^{\underline{B}} \times NF^{\underline{B}}$ such that

1. *$\approx$ Extension: $t \approx t'$ and $t' \lhd r \implies t \lhd r$*
2. *Rational Admissibility: for $x : U\underline{B} \vdash^c t : \underline{B}$*

$$\exists^\infty n.C[\mathbf{threc}^n\, x\,.\, t] \lhd r \implies C[\mathbf{threc}\, x\,.\, t] \lhd r$$

Proposition 10. *Given a (computation) approximation candidate $\lhd$ on $\underline{B}$, define its closure as the binary relation $Comps^{\underline{B}} \times Comps^{\underline{B}}$ where*

$$t \lhd^c u \iff t \approx \Omega \text{ or } (\exists r.u \Downarrow r \text{ and } t \lhd r)$$

It satisfies the following properties:

1. *Ω Property: $\Omega \lhd^c u$ for any $u \in Comps^{\underline{B}}$*
2. *$\approx$ Extension: $t \approx t'$ and $t' \lhd^c u \implies t \lhd^c u$*
3. *$\Downarrow$ Extension: $t \lhd^c u'$ and $(\forall r.u' \Downarrow r \implies u \Downarrow r) \implies t \lhd^c u$*
4. *Rational Admissibility: for $x : U\underline{B} \vdash^c t : \underline{B}$*

$$(\exists^\infty n.C[\mathbf{threc}^n x\,.\, t] \lhd^c u) \implies C[\mathbf{threc}\, x\,.\, t] \lhd^c u$$

Actions. We can then define the actions on these approximation candidates associated with each type constructor. Mostly this is done by structure (for values) or by use (for computations); the exceptions are U types and F types that we define, respectively, by structure, and by use. Note that it is the existential quantification in the definition of the F action that—very much like PCF sums—requires the use of the tests. Using them, we can easily define, by induction, the approximation relation and thereby establish the adequacy of the theory.

Proposition 11 (Thunk Action). *Let $\lhd$ be an approximation candidate for $\underline{B}$. Then the binary relation*

$$v \langle U(\lhd) \rangle w \iff \mathbf{force}\, v \lhd^c \mathit{unthunk}\, w$$

is an approximation candidate for $U\underline{B}$.

Proposition 12 (F Action). *Let $\lhd$ be an approximation candidate for A. Then the following is an approximation candidate for FA:*

$$t \langle F(\lhd) \rangle \mathbf{return}\ w \iff \exists v \lhd w. t \approx \mathbf{return}\ v$$

Definition 7 (Enviroments). *Given a typing context Γ, an* environment *σ for Γ is a substitution that maps each $x : A \in \Gamma$ to a closed term of type $\vdash^v \sigma(x) : A$. If σ_1 and σ_2 are two such, we write $\sigma_1 \lhd_\Gamma \sigma_2$ to mean $\sigma_1(x) \lhd_A \sigma_2(x)$ for all $x : A \in \Gamma$.*

Proposition 13. *For any $\Gamma \vdash^c t : B$ (resp. $\Gamma \vdash^v v : A$), and environments $\sigma_1 \lhd_\Gamma \sigma_2$ we have $t[\sigma_1] \lhd^c_{\underline{B}} t[\sigma_2]$ (resp. $v[\sigma_1] \lhd_A v[\sigma_2]$).*

Corollary 2 (Adequacy). *For any computation $\vdash^c t : \underline{B}$, if $t \Uparrow$ then $t \approx \Omega$.*

5 Polymorphic Call-by-Push-Value

Adequacy, Now For All. Our final extension deals with polymorphism. In Call-by-push-value, polymorphic types are computation types. We may quantify over both value and computation types. The extension is presented in Table 8.

We assume two disjoint countable sets of variables, $X, Y, \ldots \in \mathrm{VVars}$ and $\underline{X}, \underline{Y}, \ldots \in \mathrm{CVars}$, for value and computation types (resp.). Types are now also considered up to α-equivalence. They will also be considered under context, $\Theta \vdash^C \underline{B}$ and $\Theta \vdash^V A$, where Θ is some finite subset of $\mathrm{VVars} \cup \mathrm{CVars}$ that includes the free type variables of the A or $\underline{B}$. (These type judgements have an obvious inductive definition). The proper extension of a type context Θ by a type variable χ will be denoted by χ, Θ. Typing judgements also need to be annotated by a type context, as in $\Theta; \Gamma \vdash^c t : \underline{B}$ where Θ includes all the free type variables in the types of Γ and $\underline{B}$. The previous typing rules are extended in the evident way.

Table 8. Polymorphic Call-by-push-value with recursion

Types $A = X \in \mathrm{VVars} \mid \ldots \qquad \underline{B} = \underline{X} \in \mathrm{CVars} \mid \ldots \mid \prod X.\underline{B} \mid \prod \underline{X}.\underline{B}$

Typing

$$\frac{X, \Theta; \Gamma \vdash^c t : \underline{B} \quad (X \notin \Theta)}{\Theta; \Gamma \vdash^c \Lambda X.t : \prod X.\underline{B}} \qquad \frac{\Theta; \Gamma \vdash^c t : \prod X.\underline{B} \quad \Theta \vdash^V A}{\Theta; \Gamma \vdash^c tA : \underline{B}[A/X]}$$

$$\frac{\underline{X}, \Theta; \Gamma \vdash^c t : \underline{B} \qquad (\underline{X} \notin \Theta)}{\Theta; \Gamma \vdash^c \Lambda \underline{X}.t : \prod \underline{X}.\underline{B}} \qquad \frac{\Theta; \Gamma \vdash^c t : \prod \underline{X}.\underline{B} \quad \Theta \vdash^C \underline{B'}}{\Theta; \Gamma \vdash^c t\underline{B'} : \underline{B}[\underline{B'}/\underline{X}]}$$

Reduction

$$\frac{}{\Lambda X.t \Downarrow \Lambda X.t} \qquad \frac{t \Downarrow \Lambda X.u \quad u[A/X] \Downarrow r}{tA \Downarrow r}$$

$$\frac{}{\Lambda \underline{X}.t \Downarrow \Lambda \underline{X}.t} \qquad \frac{t \Downarrow \Lambda \underline{X}.u \quad u[\underline{B}/\underline{X}] \Downarrow r}{t\underline{B} \Downarrow r}$$

Table 9. Extension of the theory in Table 7 to polymorphism

β Rules	$(\Lambda X.t)A \approx t[A/X]$	$(\Lambda \underline{X}.t)\underline{B} \approx t[\underline{B}/\underline{X}]$
Divergence Rules	$\Omega A \approx \Omega$	$\Omega \underline{B} \approx \Omega$

Reduction and Theory. Reduction—defined only for closed terms of closed type—is still deterministic. On the theory end of things, we equate only closed terms of closed type so that we need only extend the theory of Sect. 4 with the obvious β and divergence rules (Table 9). Unsurprisingly, soundness still stands.

5.1 Adequacy

Approximation Candidates and Actions. Throughout we have worked with approximation candidates—and now we can reap the fruits of that work. The definition of approximation candidates (Definition 6) and of their extension to computations (Proposition 10) can stay exactly the same; as can the actions for non-polymorphic type constructors. The actions of polymorphic types follow.

Proposition 14. *Let $Y \vdash^C \underline{B}$ be a computation type, and ϕ a mapping that assigns to every closed type T and approximation candidate $\triangleleft \in ACs^T$ an approximation candidate $\phi_{T,\triangleleft} \in ACs^{\underline{B}[T/Y]}$; then*

$$t \left\langle \prod Y.\phi \right\rangle \Lambda Y.u \iff \textit{for all } \vdash^C T, \triangleleft \in ACs^T . tT \langle \phi_{T,\triangleleft} \rangle^c u[T/Y]$$

is an approximation candidate for $\prod Y.\underline{B}$—and likewise for $\prod \underline{Y}.\underline{B}$

Approximations. The approximation relations need to be parametrized by the candidates that will instantiate the type variables so that in the end we arrive at a candidate for a closed type. As usual, we have that it satisfies the weakening and substitution properties that are used in the proof of adequacy for abstractions and type instantiations, respectively.

Definition 8 (Approximation Environment). *An* approximation environment γ *for* Θ *is a map taking each* $\chi \in \Theta$ *to a closed type* $\gamma^T(\chi)$ *of the same kind as* χ *and an adequacy candidate* $\gamma^C(\chi) \in ACs^{\gamma^T(\chi)}$.

Definition 9 (Parametrized Approximation Relations). *Let* $\Theta \vdash^V A$ *(resp.* $\Theta \vdash^C \underline{B}$*) be a (possibly open) type and* γ *an approximation environment for* Θ*. The following* parametrized approximation relations*, defined by induction on types, determine an approximation candidate for* $A[\gamma^T]$*—i.e. A with each type variable* χ *replaced with* $\gamma^T(\chi)$ *(resp.* $\underline{B}[\gamma^T]$*).*

$$\begin{aligned}
\lhd^\gamma_{\Theta\vdash^V X} &= \gamma^C(X) & \lhd^\gamma_{\Theta\vdash^C \underline{X}} &= \gamma^C(\underline{X}) \\
\lhd^\gamma_{\Theta\vdash^V 1} &= \lhd_1 & \lhd^\gamma_{\Theta\vdash^V A\times A'} &= (\lhd^\gamma_{\Theta\vdash^V A}) \times (\lhd^\gamma_{\Theta\vdash^V A'}) \\
\lhd^\gamma_{\Theta\vdash^V 0} &= \lhd_0 & \lhd^\gamma_{\Theta\vdash^V A+A'} &= (\lhd^\gamma_{\Theta\vdash^V A}) + (\lhd^\gamma_{\Theta\vdash^V A'}) \\
\lhd^\gamma_{\Theta\vdash^V U\underline{B}} &= U(\lhd^\gamma_{\Theta\vdash^C \underline{B}}) & \lhd^\gamma_{\Theta\vdash^C FA} &= F(\lhd^\gamma_{\Theta\vdash^V A}) \\
\lhd^\gamma_{\Theta\vdash^C 1_\Pi} &= (\lhd_{1_\Pi}) & \lhd^\gamma_{\Theta\vdash^C \underline{B}\,\Pi\,\underline{B'}} &= (\lhd^\gamma_{\Theta\vdash^C \underline{B}})\ \Pi\ (\lhd^\gamma_{\Theta\vdash^C \underline{B'}}) \\
\lhd^\gamma_{\Theta\vdash^C A\to\underline{B}} &= (\lhd^\gamma_{\Theta\vdash^V A}) \to (\lhd^\gamma_{\Theta\vdash^C \underline{B}}) \\
\lhd^\gamma_{\Theta\vdash^C \prod Y.\underline{B}} &= \prod Y.\left(\lhd^{\gamma[Y\mapsto(-,=)]}_{Y,\Theta\vdash^C \underline{B}}\right\}\Big) & \lhd^\gamma_{\Theta\vdash^C \prod \underline{Y}.\underline{B}} &= \prod \underline{Y}.\left(\lhd^{\gamma[\underline{Y}\mapsto(-,=)]}_{\underline{Y},\Theta\vdash^C \underline{B}}\right\}\Big)
\end{aligned}$$

Definition 10. *For any* Θ *and approximation environment* γ *for* Θ*, if* σ_1 *and* σ_2 *are environments for* $\Gamma[\gamma^T]$*, we write* $\sigma_1 \lhd^\gamma_{\Theta;\Gamma} \sigma_2$ *to mean* $\sigma_1(x) \lhd^\gamma_{\Theta\vdash^V A} \sigma_2(x)$ *for every* $x : A \in \Gamma$.

Proposition 15. *For any* $\Theta; \Gamma \vdash^c t : B$ *(resp.* $\Theta, \Gamma \vdash^v v : A$*), approximation environment* γ *for* Θ*, and environments* $\sigma_1 \lhd^\gamma_{\Theta;\Gamma} \sigma_2$ *for* Γ

$$t[\gamma^T][\sigma_1] \left\langle \lhd^\gamma_{\Theta\vdash^C \underline{B}} \right\rangle^c t[\gamma^T][\sigma_2] \qquad \left(\textit{resp. } v[\gamma^T][\sigma_1] \left\langle \lhd^\gamma_{\Theta\vdash^V A} \right\rangle v[\gamma^T][\sigma_2]\right)$$

6 Concluding Remarks

We have thus seen how, for term-level recursion, the rational continuity rule coupled with β, the fixpoint property of recursion, and strictness of the basic constructors of the language suffices to make a theory adequate. The recipe of the previous sections applies to both call-by-name and call-by-value languages and is compatible with polymorphic types. Along the way we used no category theory; no models were mentioned. We relied only on syntactic constructions and required no external machinery.

Two extensions are conspicuous for their absence: to existential types and to recursive types. In Call-by-push-value, existential types are value types. We conjecture our theorem holds for them but we must find a way to quantify over ultimate patterns. For recursive types, even finding suitable conditions on $\approx$ is challenging. We would like to adapt Pitts' (1996) method of minimal invariant relations but we will need type constructors to be functorial over suitable syntactic categories.

For term-recursion and polymorphism, however, we now know that to prove a model adequate we need only to show that it satisfies the basic laws and rational continuity.

References

Abramsky, S., Jagadeesan, R., Malacaria, P.: Full abstraction for PCF. Inf. Comput. **163**(2), 409–470 (2000)

Abramsky, S., McCusker, G.: Call-by-value games. In: Nielsen, M., Thomas, W. (eds.) CSL 1997. LNCS, vol. 1414, pp. 1–17. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0028004

Bloom, S.L., Ésik, Z.: Iteration Theories - The Equational Logic of Iterative Processes. EATCS Monographs on Theoretical Computer Science. Springer, Heidelberg (1993). https://doi.org/10.1007/978-3-642-78034-9

Fiore, M.P., Plotkin, G.D.: An axiomatization of computationally adequate domain theoretic models of FPC. In: LICS: IEEE Symposium on Logic in Computer Science (1994)

Fiore, M.P.: Axiomatic Domain Theory in Categories of Partial Maps, Distinguished Dissertations in Computer Science, vol. 14. Cambridge University Press, Cambridge (1996)

Girard, J.Y., Lafont, Y., Taylor, P.: Proofs and Types, Cambridge Tracts in Theoretical Computer Science, vol. 7. Cambridge University Press, Cambridge (1989)

Laird, J.: Game semantics for a polymorphic programming language. J. ACM **60**(4), 29:1–29:27 (2013)

Lassen, S.B., Levy, P.B.: Typed normal form bisimulation for parametric polymorphism. In: Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24–27 June 2008, Pittsburgh, PA, USA, pp. 341–352 (2008)

Levy, P.B.: Call-By-Push-Value: A Functional/Imperative Synthesis, Semantics Structures in Computation, vol. 2. Springer, Dordrecht (2004). https://doi.org/10.1007/978-94-007-0954-6

McCusker, G.: Games and Full Abstraction for a Functional Metalanguage with Recursive Types. CPHC/BCS Distinguished Dissertations. Springer, London (1998). https://doi.org/10.1007/978-1-4471-0615-9

Møgelberg, R.E.: From parametric polymorphism to models of polymorphic FPC. Math. Struct. Comput. Sci. **19**(4), 639–686 (2009)

Normann, D.: On sequential functionals of type 3. Math. Struct. Comput. Sci. **16**(2), 279–289 (2006)

Pitts, A.M.: Relational properties of domains. Inf. Comput. **127**(2), 66–90 (1996)

Pitts, A.M.: Parametric polymorphism and operational equivalence. Math. Struct. Comput. Sci. **10**(3), 321–359 (2000)

Plotkin, G.D.: LCF considered as a programming language. Theor. Comput. Sci. **5**(3), 223–255 (1977)

Simpson, A.K.: Computational adequacy for recursive types in models of intuitionistic set theory. Ann. Pure Appl. Log. **130**(1–3), 207–275 (2004)

Wright, J.B., Thatcher, J.W., Wagner, E.G., Goguen, J.A.: Rational algebraic theories and fixed-point solutions. In: 1976 17th Annual Symposium on Foundations of Computer Science, pp. 147–158. IEEE (1976)

Zeilberger, N.: Focusing and higher-order abstract syntax. In: Proceedings of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, pp. 359–369. ACM, New York (2008)