A Metalanguage for Guarded Iteration

Sergey Goncharov[a] Christoph Rauch[a] Lutz Schröder[a]

University of Minho, Braga, July 31, 2018

[a]Friedrich-Alexander-Universität Erlangen-Nürnberg

Prelude: Guarded Iteration

Monads for Computations in One Slide

- Monads formalize generalized functions $f : X \to TY$, like nondeterministic (with $T = \mathcal{P}X$) or partial (with $TX = X + 1$)[1]

[1] Moggi 1991, Notions of Computation and Monads
[2] Plotkin and Power 2002, Notions of Computation Determine Monads

Monads for Computations in One Slide

- Monads formalize generalized functions $f : X \to TY$, like nondeterministic (with $T = \mathcal{P}X$) or partial (with $TX = X + 1$)[1]
- T is a type constructor, together with operations $\eta : X \to TX$ (unit) and $(f : X \to TY) \mapsto (f^\star : TX \to TY)$ (lifting), inducing the Klesili category of T under

  $$\mathrm{id} = \eta : X \to TX \qquad f \diamond g = (f : Y \to TZ)^\star\,(g : X \to TY)$$

  In Haskell's point-full notation: do $x \leftarrow p; f(x) = f^\star(p)$

[1] Moggi 1991, Notions of Computation and Monads
[2] Plotkin and Power 2002, Notions of Computation Determine Monads

Monads for Computations in One Slide

- Monads formalize generalized functions $f : X \to TY$, like nondeterministic (with $T = \mathcal{P}X$) or partial (with $TX = X + 1$)[1]
- T is a type constructor, together with operations $\eta : X \to TX$ (unit) and $(f : X \to TY) \mapsto (f^\star : TX \to TY)$ (lifting), inducing the Klesili category of T under

$$\mathrm{id} = \eta : X \to TX \qquad f \diamond g = (f : Y \to TZ)^\star\,(g : X \to TY)$$

  In Haskell's point-full notation: do $x \leftarrow p; f(x) = f^\star(p)$

- Duality of operations and effects[2]: e.g. for $T = \mathcal{P}$, $\mathit{toss} = \{\mathit{head}, \mathit{tail}\}$

$$p + q = \text{do } x \leftarrow \mathit{toss}; \text{if } (x = \mathit{head}) \text{ then } p \text{ else } q.$$

  In this sense T_Σ extends T with Σ-operations, e.g. for $\Sigma = A \times -$:

$$a.\,p = \text{do } (\mathit{action}_a : 1 \to T_\Sigma 1); p$$

[1] Moggi 1991, Notions of Computation and Monads
[2] Plotkin and Power 2002, Notions of Computation Determine Monads

Abstract Guardedness on Monads

Abstract guardedeness for a monad T is a relation between Kleisi morphisms $f : X \to TY$ and summands $\sigma : Y' \hookrightarrow Y$ satisfying

$$\textbf{(trv)} \quad \frac{f : X \to TY}{(T\,\mathsf{in}_1)\,f : X \to_{\mathsf{in}_2} T(Y + Z)}$$

$$\textbf{(sum)} \quad \frac{f : X \to_\sigma TZ \qquad g : Y \to_\sigma TZ}{[f, g] : X + Y \to_\sigma TZ}$$

$$\textbf{(cmp)} \quad \frac{f : X \to_{\mathsf{in}_2} T(Y + Z) \qquad g : Y \to_\sigma TV \qquad h : Z \to TV}{[g, h]^\star\, f : X \to_\sigma TV}$$

where $f : X \to_\sigma TY$, equivalently $f \in \mathsf{Hom}_\sigma(X, TY)$, means that f and σ are in the relation

Abstract Guardedness on Monads

A monad is guarded Elgot if it supports partial iteration operator sending each $f : X \to_2 T(Y + X)$ to $f^\dagger : X \to TY$ satisfying the fixpoint law

$$f^\dagger = [\eta, f^\dagger]^\star f$$

and other laws of iteration[3]

Example: $TX = (X \times Nat^\star) \cup Nat^\omega$, equivalently, TX is a final $(X + Nat \times -)$-coalgebra

TX contains

- pairs (x, τ) of a result $x \in X$ and a finite trace $\tau \in Nat^\star$, and
- infinite traces $\pi \in Nat^\omega$

[3] Bloom and Ésik 1993, Iteration theories: The equational logic of iterative processes

A Monad of (In)Finite Traces

- The unit of $TX = (X \times Nat^\star) \cup Nat^\omega$ sends x to $(x, \langle\,\rangle)$
- Given $f : X \to TY$,

$$f^\star(x,\tau) = \begin{cases} (y, \tau +\!\!+ \tau') & \text{if } f(x) = (y,\tau'), \\ \tau +\!\!+ \pi & \text{if } f(x) = \pi, \end{cases} \qquad f^\star(\pi) = \pi.$$

- $f : X \to_{\mathsf{inr}} (Y + Z) \times Nat^\star \cup Nat^\omega$ if for every $x \in X$,

$$f(x) \in Z \times Nat^\star \quad \text{implies} \quad f(x) \in Z \times Nat^+$$

- Given $f : X \to_{\mathsf{inr}} T(Y + X) = (Y + X) \times Nat^\star \cup Nat^\omega$,

$$f^\dagger(x) = \begin{cases} (y, \tau_1 +\!\!+ \cdots +\!\!+ \tau_n) & \text{if } f(x) = (\mathsf{in}_2\, x_1, \tau_1), \ldots, f(x_n) = (\mathsf{in}_1\, y, \tau_n), \\ \tau_1 +\!\!+ \cdots +\!\!+ \tau_{n-1} +\!\!+ \pi & \text{if } f(x) = (\mathsf{in}_2\, x_1, \tau_1), \ldots, f(x_n) = \pi, \\ \tau_1 +\!\!+ \tau_2 +\!\!+ \cdots & \text{if } f(x) = (\mathsf{in}_2\, x_1, \tau_1), f(x_1) = (\mathsf{in}_1\, x_2, \tau_2), \ldots \end{cases}$$

The Metalanguage

Metalanguages for (Guarded) Iteration: Motivation

- Guardedness is a fundamental notion: Just like Moggi's computational metalanguage is a metalanguage of abstract effects, the metalanguage for guarded iteration is a metalanguage of abstract guardedness
- The metalangauge for guarded iteration can be used as a 'core programming language' for effects associated with monads. The stock of examples is growing: various process semantic domains, hybrid monads, etc.

The Main Idea

Geron and Levy[4] observed that

- modelling iteration directly would amount to syntax like

$$\text{return}\,\text{inr}\ldots\text{inr}\,\text{inl}\ldots$$

  which is like using De Bruijn indexes instead of variables
- they also proposed to use labels to index coproduct summands in $f : X \to T(\sum_i X_i)$, so as to be able to point the branch in which to iterate

Here, we assume labels = exceptions, for they can be uniformly used in three constructs

exception raising	exception handling	iteration
$\text{raise}_e\, v$	handle x in p with q	handleit $x = v$ in p

[4] Geron and Levy 2016, Iteration and labelled iteration

Quick Example

```
handleit e = ⋆ in
  handle u in
    (print ("think of a number") & raise_u ⋆)
  with
    (do y ← random();
        z ← read();
        if (y = z) then ret ⋆ else raise_e ⋆)
```

Syntax

Types:

$$A, B, \ldots ::= C \mid 0 \mid 1 \mid A + B \mid A \times B \qquad (C \in \text{Base})$$

Signatures:

- value signature Σ_v of $f : A \to B$ (e.g. $+ : Nat \times Nat \to Nat$)
- effect signature Σ_c of $f : A \to B[C]$ (e.g. put : $Nat \to 0[1]$)

Value and Computation Term Judgements:

$$\Gamma \vdash_{\mathsf{v}} v : A \qquad \text{and} \qquad \Delta \mid \Gamma \vdash_{\mathsf{c}} p : A$$

where

$$\Gamma = (x_1 : A_1, \ldots, x_1 : A_1) \qquad \text{(variable context)}$$
$$\Delta = (e_1 : E_1^{\alpha_1}, \ldots, e_m : E_m^{\alpha_m}) \qquad \text{(exception context)}$$

and $\alpha_i \in \{g, u\}$ indicate (un-)guardedness

Some Derivation Rules

$$\frac{e : E^{\mathsf{g}} \text{ in } \Delta \qquad f : A \to 0[1] \in \Sigma_c \qquad \Gamma \vdash_{\mathsf{v}} p : A \qquad \Gamma \vdash_{\mathsf{v}} q : E}{\Delta \mid \Gamma \vdash_{\mathsf{c}} f(p) \,\&\, \mathsf{raise}_e\, q : D}$$

$$\frac{\Delta, e : E^{\mathsf{g}} \mid \Gamma \vdash_{\mathsf{c}} p : A \qquad \Delta' \mid \Gamma, e : E \vdash_{\mathsf{c}} q : A \qquad |\Delta| = |\Delta'|}{\Delta \mid \Gamma \vdash_{\mathsf{c}} \mathsf{handle}\ e\ \mathsf{in}\ p\ \mathsf{with}\ q : A}$$

$$\frac{e : E^{\mathsf{u}} \text{ in } \Delta \qquad \Gamma \vdash_{\mathsf{v}} q : E}{\Delta \mid \Gamma \vdash_{\mathsf{c}} \mathsf{raise}_e\, q : D}$$

$$\frac{\Gamma \vdash_{\mathsf{v}} p : E \qquad \Delta, e : E^{\mathsf{g}} \mid \Gamma, e : E \vdash_{\mathsf{c}} q : A}{\Delta \mid \Gamma \vdash_{\mathsf{c}} \mathsf{handleit}\ e = p\ \mathsf{in}\ q : A}$$

Generic Denotational Semantics

Typing the Semantics

Types:

$$\underline{0} = \emptyset, \qquad \underline{1} = 1, \qquad \underline{A + B} = \underline{A} + \underline{B}, \qquad \underline{A \times B} = \underline{A} \times \underline{B}.$$

$$\underline{\Gamma} = \underline{A_1} \times \ldots \times \underline{A_n} \qquad \text{for } \Gamma = (x_1 : A_1, \ldots, x_n : A_n)$$

$$\underline{\Delta} = \underline{E_1} + \ldots + \underline{E_m} \qquad \text{for } \Delta = (e_1 : E_1^{\alpha_1}, \ldots, e_m : E_m^{\alpha_m})$$

Signatures:

$$[\![f]\!] \in \mathrm{Hom}(\underline{A}, \underline{B}) \qquad \text{for} \qquad f : A \to B \in \Sigma_v$$

$$[\![f]\!] \in \mathrm{Hom}_{\mathsf{inr}}(\underline{A}, T(\underline{B} + \underline{C})) \qquad \text{for} \qquad f : A \to B[C] \in \Sigma_c$$

Terms:

$$[\![\Gamma \vdash_{\mathsf{v}} v : A]\!] \in \mathrm{Hom}(\underline{\Gamma}, \underline{A}) \qquad [\![\Delta \mid \Gamma \vdash_{\mathsf{c}} p : A]\!] \in \mathrm{Hom}_{!+\sigma_\Delta}(\underline{\Gamma}, T(\underline{A} + \underline{\Delta}))$$

where $\sigma_\Delta : \underline{\Delta}' \hookrightarrow \underline{\Delta}$ corresponds to removal of all unguarded exceptions $e : E^u$ from Δ

Assumptions on the Model

The underlying model consists of a category $\mathbf{C}$ and a monad $\mathbb{T}$ on $\mathbf{C}$, such that

- $\mathbf{C}$ is distributive, i.e. essentially supports a distributivity isomorphism $\mathrm{dist} : A \times (B + C) \cong A \times B + A \times C$
- $\mathbb{T}$ is strong, i.e. equipped with tensorial strength $\tau_{A,B} : A \times TB \to T(A \times B)$
- $\mathbb{T}$ supports guarded iteration, and additionally validated the rule

$$\textbf{(str)} \quad \frac{f : X \to_\sigma TY}{\tau\,(\mathrm{id}_Z \times f) : Z \times X \to_{\mathrm{id}\times\sigma} T(Z \times Y)}$$

(this yields strong iteration:

$$\frac{f : W \times X \to_{\mathrm{inr}} T(Y + X)}{f^\ddagger = (T(\mathrm{snd} + \mathrm{id})\,(T\,\mathrm{dist})\tau\langle \mathrm{fst}, f\rangle)^\dagger : W \times X \to TY}\ )$$

- Function spaces $A \to_\Delta B$ can be added at little cost

Operational Semantics and Adequacy

A Monad of (In)finite Traces

Geron and Levy[5] elaborated the maybe monad – $+1$ on **Set** as the simplest monad for unguarded iteration. Incidentally, it is an initial Elgot monad on **Set**[6]

We elaborate $TX = (X \times Nat^\star) \cup Nat^\omega$ as the simplest monad for properly guarded iteration on **Set**.

- The only base type is Nat
- Value signature contains arithmetic operations
- Effect signature contains only $put : Nat \to 0[1]$

[5] Geron and Levy 2016, Iteration and labelled iteration

[6] Goncharov, Rauch, and Schröder 2015, Unguarded recursion on coinductive resumptions

Big-Step Operational Semantics

Values, Computations, Terminals:

$$v, w ::= x \mid \star \mid 0 \mid \mathit{succ}\, v \mid \mathsf{inl}\, v \mid \mathsf{inr}\, v \mid \langle v, w \rangle \mid \dots$$

$$p, q ::= \mathsf{ret}\, v \mid \mathit{pred}\, v \mid \mathit{put}\, v \mid \mathsf{raise}_x\, v \mid \mathit{put}\, v \,\&\, \mathsf{raise}_x\, w \mid \dots$$

$$t ::= \mathsf{ret}\, v, \tau \mid \mathsf{raise}_x\, v, \tau \mid \pi \qquad (\tau \in \mathit{Nat}^\star, \pi \in \mathit{Nat}^\omega)$$

Some Rules:

$$\frac{}{\mathit{put}\, v \,\&\, \mathsf{raise}_x\, w \Downarrow \mathsf{raise}_x\, w, \langle v \rangle}$$

$$\frac{v_0 = v \quad q[v_0/x] \Downarrow \mathsf{raise}_x\, v_1, \tau_1 \qquad \dots \qquad q[v_{n-1}/x] \Downarrow t, \tau_n}{\mathsf{handleit}\ x = v\ \mathsf{in}\ q \Downarrow t, \tau_1 +\!\!+ \cdots +\!\!+ \tau_n}$$

$$\frac{v_0 = v \qquad q[v_0/x] \Downarrow \mathsf{raise}_x\, v_1, \tau_1 \quad \dots \quad q[v_{n-1}/x] \Downarrow \pi}{\mathsf{handleit}\ x = p\ \mathsf{in}\ q \Downarrow \tau_1 +\!\!+ \cdots +\!\!+ \tau_{n-1} +\!\!+ \pi}$$

$$\frac{v_0 = v \qquad q[v_0/x] \Downarrow \mathsf{raise}_x\, v_1, \tau_1 \qquad q[v_1/x] \Downarrow \mathsf{raise}_x\, v_2, \tau_2 \qquad \dots}{\mathsf{handleit}\ x = p\ \mathsf{in}\ q \Downarrow \tau_1 +\!\!+ \tau_2 +\!\!+ \cdots}$$

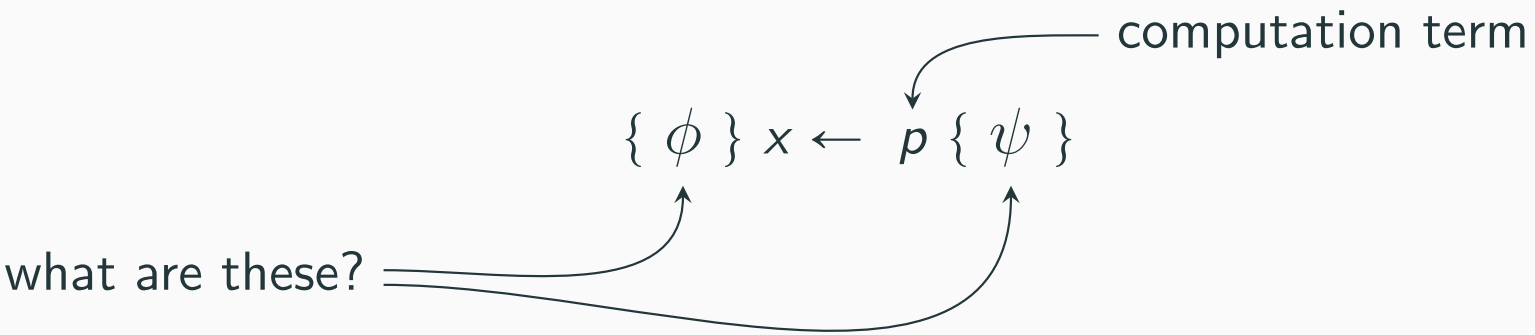
The Adequacy Theorem

Theorem (Adequacy): Let $\Delta \mid - \vdash_{\mathsf{c}} p : B$. Then,

1. If $p \Downarrow \mathsf{ret}\, v, \tau$ then $[\![\Delta \mid - \vdash_{\mathsf{c}} p : B]\!] = (\mathsf{in}_1\, v, \tau) \in (B + \Delta) \times \mathit{Nat}^\star$
2. If $p \Downarrow \mathsf{raise}_x\, v, \tau$ and $x : E^{\mathsf{g}}$ is in Δ then
   $[\![\Delta \mid - \vdash_{\mathsf{c}} p : B]\!] = (\mathsf{in}_2\, \mathsf{in}_x\, v, \tau) \in (B + \Delta) \times \mathit{Nat}^+$
3. If $p \Downarrow \mathsf{raise}_x\, v, \tau$ and $x : E^{\mathsf{u}}$ is in Δ then
   $[\![\Delta \mid - \vdash_{\mathsf{c}} p : B]\!] = (\mathsf{in}_2\, \mathsf{in}_x\, v, \tau) \in (B + \Delta) \times \mathit{Nat}^\star$
4. If $p \Downarrow \pi$, then $[\![\Delta \mid - \vdash_{\mathsf{c}} p : B]\!] = \pi \in \mathit{Nat}^\omega$

Conclusions & Further Work

- The metalanguage for guarded iteration provides an extensible platform for programming with guarded iteration
- More concrete monads ⇒ more concrete operational semantics and more adequacy theorems
- Further case study: a monad for hybrid computation with guardedness as progressiveness[7]
- Hoare logic for guarded iteration:



[7] Goncharov, Jakob, and Neves 2018, A Semantics for Hybrid Iteration

References

Stephen L. Bloom and Zoltán Ésik. *Iteration theories: the equational logic of iterative processes*. Springer-Verlag New York, Inc., New York, NY, USA, 1993.

Bram Geron and Paul Blain Levy. Iteration and labelled iteration. In *Mathematical Foundations of Programming Semantics, MFPS XXXII*, volume 325, pages 127 – 146, 2016.

Sergey Goncharov, Christoph Rauch, and Lutz Schröder. Unguarded recursion on coinductive resumptions. In *Mathematical Foundations of Programming Semantics, MFPS 2015*, 2015.

References II

Sergey Goncharov, Julian Jakob, and Renato Neves. A semantics for hybrid iteration. In Sven Schewe and Lijun Zhang, editors, *29th International Conference on Concurrency Theory (CONCUR 2018)*, 2018.

Eugenio Moggi. Notions of computation and monads. *Inf. Comput.*, 93: 55–92, 1991.

Gordon Plotkin and John Power. Notions of computation determine monads. In *FoSSaCS'02*, volume 2303, pages 342–356, 2002.