A Metalanguage for Guarded Iteration

Christoph Rauch

TCS Seminar SS 2018

FAU Erlangen-Nürnberg

June 19, 2018

Computations from different perspectives

Domain theory

- computations are identified with final result (if any)
- programs either terminate with a value, or they diverge
- *extensional* paradigm

Computations from different perspectives

Domain theory

- computations are identified with final result (if any)
- programs either terminate with a value, or they diverge
- *extensional* paradigm

Process algebra

- computations are processes unfolding in time
- main attribute: behaviour (of an infinitely running process)
- *intensional* paradigm

Guardedness

General idea

ensure progress or productivity in (co)recursive definitions

In process algebra

- recursive process specification $X = t$ guarded if every occurrence of X in t is under an action
- e.g. in *CCS* under bisimulation semantics: guarded recursive specifications have unique solutions [Milner '83]
- for example,

$$P = a.P$$

keeps performing the action a

Guardedness

General idea

ensure progress or productivity in (co)recursive definitions

More recently: in (co)programming

- guardedness analysis in Coq for corecursive definitions, proofs by corecursion:

  are corecursive calls (directly!) under constructors?

- guarded recursion by typing/functorial guardedness
  [Birkedal, Møgelberg '13],[Vezzosi '15],[Milius, Litak '13], [Clouston et al. '15] and others

(Abstractly) Guarded co-Cartesian Categories

Inference rules

$$\frac{f : X \to Y}{\mathsf{in}_1\, f : X \to_{\mathsf{in}_2} Y + Z} \qquad \frac{f : X \to_\sigma Z \quad g : Y \to_\sigma Z}{[f, g] : X + Y \to_\sigma Z}$$

$$\frac{f : X \to_{\mathsf{in}_2} Y + Z \qquad g : Y \to_\sigma V \qquad h : Z \to V}{[g, h] \circ f : X \to_\sigma V}$$

Definition (Guarded co-Cartesian category)

A co-Cartesian category $\mathbf{C}$ equipped with distinguished subsets $\mathrm{Hom}_\sigma(X, Y) \subseteq \mathrm{Hom}(X, Y)$ of *partially guarded morphisms* for $A, B \in |\mathbf{C}|$, and any summand $\sigma : Y_1 \to Y_1 + Y_2 \simeq Y$ satisfying the rules above is called *guarded* ($f : X \to_\sigma Y$ means $f \in \mathrm{Hom}_\sigma(X, Y)$).

Guardedness for Monads

Abstract guardedness for a strong monad T

- guarded: Kleisli category of T is a guarded category
- $$\frac{f : X \to_\sigma TY}{\tau \circ \langle \mathsf{id}, f \rangle : X \to_{\mathsf{id} \times \sigma} T(X \times Y)}$$
- underlying category distributive
- guarded pre-iterative: every $\sigma +$ id-guarded morphism has a solution w.r.t.

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

  (and $f^\dagger$ is σ-guarded)

Guardedness for Monads

Examples

- Finitely branching processes: $TX = \nu\gamma.\, \mathcal{P}_\omega(X + \mathrm{Act} \times \gamma)$
- $TX = \mathcal{P}X$ admits total iteration
- $TX = (\nu\gamma.\, X + S \times \gamma)^S$, stateful traces

Guardedness for Monads

Examples

- Finitely branching processes: $TX = \nu\gamma.\, \mathcal{P}_\omega(X + \mathrm{Act} \times \gamma)$
- $TX = \mathcal{P}X$ admits total iteration
- $TX = (\nu\gamma.\, X + S \times \gamma)^S$, stateful traces

Metalanguages

- Moggi: metalanguage for strong monads to model effectful programs
- Here: metalanguage for strong guarded (pre-)iterative monads to model effectful programs with guarded iteration

Setup

Labelled iteration

- imperative languages: break/continue for loops
- [Geron, Levy '16]: enable break/continue in Fine-Grain Call-By-Value
- modeled using exceptions
- guardedness: also indicated via coproducts; similar idea can be used
- organise guarded control flow using labels/exceptions

Fine-Grain Call-By-Value [Levy et al. '02]

- separate judgements for values and computations
- more suitable for operational semantics than Moggi's *monadic metalanguage*

Language overview

Design

- parametric in a guarded pre-iterative monad T
- rules for *guardedness* mirrored in the calculus
- exception model suggests breaking up composition into
  - (unguarded) composition
  - (guarded) "exception handling"
- guarded iteration construct

Language overview

Implementation

- signature containing the operations of the monad: $[\![f : A \to B[C]]\!] : A \to_2 T(B + C)$
- *two* contexts: Δ for guardedness information, Γ for program variables
- Δ stores typed "exception" names tagged with guardedness info ("g" or "u" added to the type)
- value judgements: $[\![\Gamma \vdash_v v : A]\!] : \Gamma \to A$
- computation judgements: $[\![\Delta, \Gamma \vdash_c p : B]\!] : \Gamma \to_{!+\sigma_\Delta} T(B + \Delta)$

Program example

Guessing numbers

handleit $e = *$ in
  handle u in
    $(print\ (\texttt{"think of a number"})\ \&\ raise\_u\ *\ )$
  with
    (do $y \leftarrow random();$
        $z \leftarrow read();$
        if $(y = z)$ then ret $*$ else $raise\_e\ *)$

- handle: guarded "composition"
- handleit: iteration construct
- $random$, $read$: unguarded signature operations
- $print$: guarded operation (accompanied by $raise\_u$)

Language Features

Contexts

- we want to work with multiple variables
- definable *strong iteration* operator

$$\frac{f : W \times X \to_2 T(Y + X)}{f^\ddagger = (T((\mathsf{pr}_2 + \mathsf{id}) \circ \mathsf{dist}) \circ \tau \circ \langle \mathsf{pr}_1, f \rangle)^\dagger : W \times X \to TY}$$

- . . . which is iteration in the co-Kleisli category $\mathbf{C}[W]$ of the context comonad $W \times -$

Theorem

Guarded pre-iterative monads on $\mathbf{C}$ *extend to guarded pre-iterative monads on* $\mathbf{C}[W]$ *under the same definition of guardedness and iteration defined as above.*

Language Features

Algebraic operations via Generic effects

- symbols $f : A \to B[0]$ correspond to Kleisli morphisms $A \to TB$
- for A, B natural numbers n, m, also called *generic effects*
- *algebraic operations*: certain natural transformations $T^m \to T^n$
- e.g. nondeterministic choice $\oplus : T^2 \to T^1$
- correspondence via (eliding isomorphisms):

$$TX^m \xrightarrow{\alpha} TX^n \quad \Rightarrow \quad f = \alpha(m \xrightarrow{\eta} Tm) : n \to Tm$$

$$n \xrightarrow{f} Tm \quad \Rightarrow \quad \alpha(g : m \to TX) = g^\star \circ f : v \to TX$$

Language Features

Explicit guards

- given $f : A \to B[C]$:

  $\mathsf{gcase}\, f(v)\ \mathsf{of}\ \mathsf{inl}\, x \mapsto p;\ \mathsf{inr}\, y \mapsto \mathsf{raise}_e\, q$
- raise a *guarded* exception, i.e.
  - call a guard f from the signature
  - run p if we end up in the unguarded part
  - otherwise raise e
- e^g is added to Δ

Language Features

Explicit guards

- given $f : A \to B[C]$:

  $\mathsf{gcase}\, f\,(v)\ \mathsf{of}\ \mathsf{inl}\, x \mapsto p;\ \mathsf{inr}\, y \mapsto \mathsf{raise}_e\, q$
- raise a *guarded* exception, i.e.
  - call a guard f from the signature
  - run p if we end up in the unguarded part
  - otherwise raise e
- e^g is added to Δ

Unguarded exceptions

- $\mathsf{raise}_e\, v$
- adds ordinary exceptions e^u to Δ

Language Features

Exception handling

- handle e in p with q
- currently handles only *guarded* exceptions e occurring in p
- q acts as an exception handler

Language Features

Exception handling

- handle e in p with q
- currently handles only *guarded* exceptions e occurring in p
- q acts as an exception handler

Iteration

- intended as iterated exception handling
- handleit $x = v$ in p
- iteratively runs p (depending on x, which is initialised to v)

Language Features

λ-abstraction and application

- abstraction turns a computation into a value
- values do not have guardedness annotation
- we store Δ in the type:

$$\frac{\Delta \mid \Gamma, x : A \vdash_c p : B}{\Gamma \vdash_v \lambda x.\, p : A \to_\Delta B} \qquad \frac{\Gamma \vdash_v w : A \qquad \Gamma \vdash_v v : A \to_\Delta B}{\Delta \mid \Gamma \vdash_c v\, w : B}$$

Denotational Semantics

syntactic feature	semantic equivalent
composition, handling	Kleisli composition + guardedness axioms
iterated handling	strong iteration operator
λ-abstraction, application	???

Denotational Semantics

syntactic feature	semantic equivalent
composition, handling	Kleisli composition + guardedness axioms
iterated handling	strong iteration operator
λ-abstraction, application	???

- the latter are problematic: what is $[\![A \to_\Delta B]\!]$?
- need to assume additional properties of our category ...

Greatest σ-algebras

Representable guarded homs

Assume that the contravariant functors $X \mapsto \mathsf{Hom}_\sigma(X, TA)$ are representable, i.e. there exist $A_\sigma \in |\mathbf{C}|$ and natural isomorphisms

$$\xi : \mathsf{Hom}(X, A_\sigma) \simeq \mathsf{Hom}_\sigma(X, TA)$$

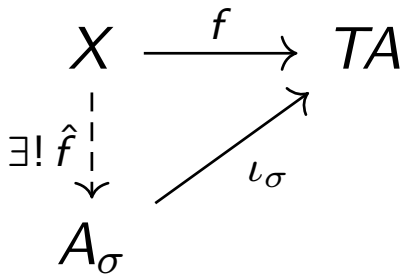
Greatest σ-algebras

Representable guarded homs

Assume that the contravariant functors $X \mapsto \mathrm{Hom}_\sigma(X, TA)$ are representable, i.e. there exist $A_\sigma \in |\mathbf{C}|$ and natural isomorphisms

$$\xi : \mathrm{Hom}(X, A_\sigma) \simeq \mathrm{Hom}_\sigma(X, TA)$$

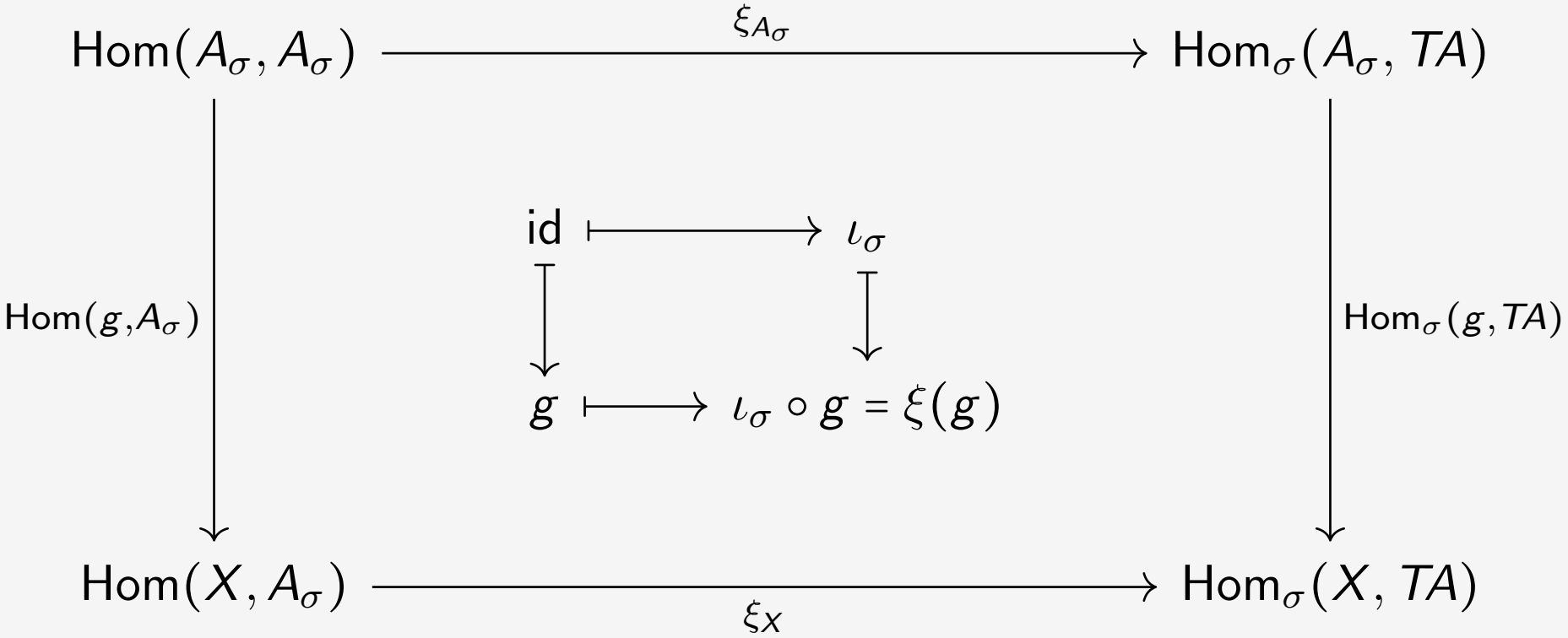
Equivalently:



(A_σ, ι_σ) *greatest* σ*-algebra*

where $f \in \mathrm{Hom}_\sigma(X, TA)$, $\iota_\sigma \in \mathrm{Hom}_\sigma(A_\sigma, TA)$

Proof


$\mathsf{Hom}(A_\sigma, A_\sigma)$
ξ_{A_σ}
$\mathsf{Hom}_\sigma(A_\sigma, TA)$
$\mathsf{Hom}(g, A_\sigma)$
$\mathsf{Hom}_\sigma(g, TA)$
id
ι_σ
g
$\iota_\sigma \circ g = \xi(g)$
$\mathsf{Hom}(X, A_\sigma)$
ξ_X
$\mathsf{Hom}_\sigma(X, TA)$

Greatest σ-algebras

Proposition

Suppose that (A_σ, ι_σ) *is a greatest* σ*-algebra. Then there is a unique* $\alpha_\sigma : TA_\sigma \to A_\sigma$ *such that* $\iota_\sigma \alpha_\sigma = \iota_\sigma^\star$. *The pair* $(A_\sigma, \alpha_\sigma)$ *is a* T*-subalgebra of* (TA, μ).

Greatest σ-algebras

Proposition

Suppose that (A_σ, ι_σ) is a greatest σ-algebra. Then there is a unique $\alpha_\sigma : TA_\sigma \to A_\sigma$ such that $\iota_\sigma \alpha_\sigma = \iota_\sigma^\star$. The pair $(A_\sigma, \alpha_\sigma)$ is a T-subalgebra of (TA, μ).

Example: $T_\Sigma X = \nu\gamma.\, T(X + \Sigma\gamma)$

For $\sigma : A' \hookrightarrow A$, $\overline{\sigma} : A'' \hookrightarrow A$, we have

$$A_\sigma = T(A'' + \Sigma T_\Sigma A) \qquad \iota_\sigma = \mathrm{out}^{-1} \circ T(\overline{\sigma} + \mathrm{id})$$

Recall that a morphism $f : X \to T_\Sigma A$ is σ-guarded iff it factors as $f = \mathrm{out}^{-1} \circ T(\overline{\sigma} + \mathrm{id}) \circ u$ for some u.

Semantics of λ-abstraction

Abstraction

Given $f := [\![\Delta \mid \Gamma, x : A \vdash_c p : B]\!]$, we let

$$[\![\Gamma \vdash_v \lambda x.\, t : A \to_\Delta B]\!] = \mathrm{curry}(\hat{f}).$$

Application

Given $g := [\![\Gamma \vdash_v v : A \to_\sigma B]\!]$ as well as some $w := [\![\Gamma \vdash_v w : A]\!]$, we let

$$[\![\Delta \mid \Gamma \vdash_c v\, w : B]\!] := \iota_\sigma \circ \mathrm{uncurry}(g) \circ \langle \mathrm{id}, w \rangle.$$

Operational Semantics

Simple trace example

- $TX = X \times \mathbb{N}^\star + \mathbb{N}^\omega$
- operation $put : \mathbb{N} \to 0[1]$ adds a number to the *trace*
- gcase construct simplifies to

$$\mathsf{gcase}\, put(n)\ \mathsf{of}\ \mathsf{inl}\, x \mapsto \mathsf{init}\, x;\ \mathsf{inr}\, \_ \mapsto \mathsf{raise}_e\, v =: put(n)\ \&\ \mathsf{raise}_e\, v$$

- guardedness: at least one output operation performed
- operationally, a (guarded!) program can either
  - terminate with a value and a finite trace
  - produce an infinite trace

Operational Semantics

Exemplary rules

$$\frac{p \Downarrow \pi}{\text{handle } x \text{ in } p \text{ with } q \Downarrow \pi}$$

$$\frac{p \Downarrow \text{raise}_x\, v, \tau \quad q[v/x] \Downarrow \pi}{\text{handle } x \text{ in } p \text{ with } q \Downarrow \tau +\!\!+ \pi}$$

$$\frac{p \Downarrow \text{ret}\, v, \tau}{\text{handle } x \text{ in } p \text{ with } q \Downarrow \text{ret}\, v, \tau}$$

$$\frac{p \Downarrow \text{raise}_y\, v, \tau}{\text{handle } x \text{ in } p \text{ with } q \Downarrow \text{raise}_y\, v, \tau}$$

where τ denotes a finite trace, π an infinite trace

Operational Semantics

Theorem (Adequacy)

Let $\Delta \mid - \vdash_c p : B$. *Then,*

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

Related and Further Work

- Adequacy result generalises [Geron, Levy '16]
- [Nakata, Uustalu '09] give a similar semantics for a WHILE language
- $M_S X = (\nu\gamma.\, X + S \times \gamma)^S$ provides a denotational background for op.cit.
- M_S is guarded pre-iterative
- how does our calculus, instantiated with M_S, relate to [Nakata, Uustalu '09]?

References I


R. Milner (TCS '83)
Calculi for synchrony and asynchrony


R. Clouston, A. Bizjak, H. B. Grathwohl, L. Birkedal (FoSSaCS '15)
Programming and Reasoning with Guarded Recursion for Coinductive Types


P. Levy, B. Geron (ENTCS '16)
Iteration and Labelled Iteration


S. Goncharov, L. Schröder, C. Rauch, M. Piróg (FoSSaCS '17)
Unifying Guarded and Unguarded Iteration


S. Goncharov, L. Schröder (FoSSaCS '18)
Guarded Traced Categories


P. Levy, J. Power, H. Thielecke (IaC '03)
Modelling Environments in Call-by-Value Programming Languages

References II

S. Milius, T. Litak (FICS '13)
Guard Your Daggers and Traces

K. Nakata, T. Uustalu (TPHOLs '09)
Trace-Based Coinductive Operational Semantics for While

L. Birkedal, R. E. Møgelberg (LICS '13)
Intensional type theory with guarded recursive types qua fixed points on universes

A. Vezzosi (PhD Thesis, 2015)
Guarded Recursive Types in Type Theory