From Delay Monad to Duration Monad: An Excursion into Non-Inductive Semantics

Sergey Goncharov

Oberseminar, TCS FAU, November 13, 2018
In this talk, no ..
Inductive Semantics

Traditional (programming language) semantics is inductive:

- Evaluation relations of **big-step semantics** are derived in rule-based systems in a finite (inductive) way
- Denotations of fixpoints are subject to Scott semantics and interpreted as **least fixpoints**
- **Hoare logic** is concerned with **partial assertions**

This paradigm has a number of implications

- **Single notion of divergence** $\perp$
- Bounded nondeterminism
- $\omega$ is the only computationally meaningful infinite ordinal $\Rightarrow$ no infinite traces
- ...
One the other extreme, co-inductive semantics flips it upside-down:

- Denotational domains are coalgebras
- Big-step operational semantics is defined in non-well-founded proof systems

Downsides:

- Computations become total, but operations on computations become partial, e.g. fixpoints, which require productivity analysis
- Semantics is usually too fine, e.g. \( \{x := 1; x := 2\} \not\sim \{x := 2\} \)
Plan of Talk

A simple while-language with inductive semantics

Non-inductive semantics via the delay monad

A call-by-value while-language for hybrid computation

The duration monad

An adequate duration semantics
A Sample of Inductive Semantics
Consider a simple while-language:

- **types** $A, B, \ldots$ by the grammar

  $$A, B, \ldots ::= \mathbb{N} \mid 1 \mid 2 \mid A \times B$$

- **value and computation judgments**

  $$\Gamma \vdash_v v : A \quad \text{and} \quad \Gamma \vdash_c p : A$$

  (thus, we piggyback on *fine-grain call-by-value*\(^1\))

- A signature of $\Sigma$ operations $f : A \to B$ with atomic $B$

---

\(^1\)Levy, Power, and Thielecke 2002, Modelling Environments in Call-By-Value Programming Languages
Term Formation

Values:

\[ \begin{align*}
\Gamma & \vdash_{v} \star : 1 \\
\Gamma & \vdash_{v} x : A \\
\Gamma & \vdash_{v} \text{true} : 2 \\
\Gamma & \vdash_{v} \text{false} : 2 \\
\Gamma & \vdash_{v} f(v) : B \\
\Gamma & \vdash_{v} \langle v, w \rangle : A \times B
\end{align*} \]

Computations:

\[ \begin{align*}
\Gamma & \vdash_{c} p : A \\
\Gamma & \vdash_{c} x : A \\
\Gamma, x : A & \vdash_{c} q : B \\
\Gamma & \vdash_{c} x := p; q : B \\
\Gamma & \vdash_{v} v : A \\
\Gamma & \vdash_{c} [v] : A \\
\Gamma & \vdash_{c} \text{if } v \text{ then } p \text{ else } q : A \\
\Gamma & \vdash_{c} x : A \\
\Gamma, x : A & \vdash_{v} v : 2 \\
\Gamma & \vdash_{c} p : A \\
\Gamma, x : A & \vdash_{c} q : A \\
\Gamma & \vdash_{c} \langle x, y \rangle := v; p : C \\
\Gamma & \vdash_{c} \text{while } v \{ q \} : A \\
\Gamma & \vdash_{c} p : A \\
\Gamma, x : A & \vdash_{v} v : 2 \\
\Gamma & \vdash_{c} p : A \\
\Gamma, x : A & \vdash_{c} q : A
\end{align*} \]
Small-Step Operational Semantics

Closed values, Closed computations:

\[ v, w ::= x \mid \ast \mid \text{true} \mid \text{false} \mid \langle v, w \rangle \mid f(v) \quad (f \in \Sigma) \]

\[ p, q ::= \langle x, y \rangle ::= \langle v, w \rangle; p \mid \text{if } v \text{ then } p \text{ else } q \mid [v] \]

\[ x ::= p; q \mid x ::= p \text{ while } v \{ q \} \]

Rules:

\[ \langle x, y \rangle ::= \langle v, w \rangle; q \rightarrow q[v/x, w/y] \]

\[ x ::= [v]; q \rightarrow q[v/x] \quad \text{if true then } p \text{ else } q \rightarrow p \quad \text{if false then } p \text{ else } q \rightarrow q \]

\[ p \rightarrow p' \]

\[ x ::= p \text{ while } v \{ q \} \rightarrow x ::= p' \text{ while } v \{ q \} \]

\[ w[v/x] = \text{false} \]

\[ x ::= [v] \text{ while } w \{ q \} \rightarrow [v] \]

\[ w[v/x] = \text{true} \]

\[ x ::= [v] \text{ while } w \{ q \} \rightarrow x ::= q[v/x] \text{ while } w \{ q \} \]
Big-Step Operational Semantics

\[
\begin{align*}
[v] & \Downarrow v \\
\frac{p \Downarrow v \quad q[v/x] \Downarrow w}{x := p; q \Downarrow w} \\
\frac{p[v/x, w/y] \Downarrow u}{\langle x, y \rangle := \langle v, w \rangle; \ p \Downarrow u}
\end{align*}
\]

\[
\begin{align*}
p \Downarrow v \\
\text{if true then } p \text{ else } q \Downarrow v
\end{align*}
\]

\[
\begin{align*}
q \Downarrow v \\
\text{if false then } p \text{ else } q \Downarrow v
\end{align*}
\]

\[
\begin{align*}
p \Downarrow w \\
v[w/x] = \text{false} \\
x := p \text{ while } v \{q\} \Downarrow w
\end{align*}
\]

\[
\begin{align*}
p \Downarrow w \\
v[w/x] = \text{true} \\
x := q[w/x] \text{ while } v \{q\} \Downarrow u \\
x := p \text{ while } v \{q\} \Downarrow u
\end{align*}
\]
Equivalence of Semantics and Determinacy

**Proposition:** Big-step and small-step semantics are equivalent as follows:

\[ p \downarrow w \iff p \rightarrow^* [w] \]

**Proof Idea:** for the \((\Rightarrow)\) direction, induction over the derivation of \(p \downarrow w\); for the \((\Rightarrow)\) direction, prove

**Lemma:** \( p \rightarrow q \) with \( q \downarrow w \) imply \( p \downarrow w \)

.. and then induction over the length of \( p \rightarrow^* [w] \)

**Proposition (Determinacy):** \( p \downarrow v \) for at most one \( v \)
Assuming $\Gamma = (x_1 : A_1, \ldots, x_n : A_n)$, we interpret

$$\llbracket \Gamma \vdash v : A \rrbracket \in \text{Hom}(A_1 \times \ldots \times A_n, A)$$

$$\llbracket \Gamma \vdash p : A \rrbracket \in \text{Hom}(A_1 \times \ldots \times A_n, TA)$$

where $T$ is the maybe monad $TX = X \cup \{\bot\}$

Moreover, e.g.

$$\llbracket \Gamma \vdash x := p; q : B \rrbracket = \lambda \bar{x}. u^*(\bar{x}, h(\bar{x}))$$

$$\llbracket \Gamma \vdash [v] : A \rrbracket = \eta h$$

$$\llbracket \Gamma \vdash x := p \text{ while } v \{q\} : A \rrbracket =$$

$$\lambda \bar{x}.((\lambda x. \text{ if } b(\bar{x}, x) \text{ then } \text{inr} l(\bar{x}, x) \text{ else } \eta(\text{inl}(x)))^+) \ast (h(\bar{x}))$$

using the fact that $T$ is order-enriched, hence a (complete) Elgot monad

Elgot iteration: $(f : X \to T(Y + X)) \mapsto (f^+ : X \to TY)$
Soundness and Adequacy

Proposition (Soundness): if \( p \downarrow v \) then \( \llbracket \not\vdash c p : A \rrbracket = v \)

Proof Idea: induction over the derivation of \( p \downarrow v \)

Proposition (Computational Adequacy):
if \( \llbracket \not\vdash c p : A \rrbracket = v \) then \( p \downarrow v \)

Proof Idea: In this case easily by induction over the denotational semantic rules. Way more powerful standard tool: logical relations
Delay Monad and Its Quotient
The Delay Monad

In has been noticed early on that the maybe monad $TX = X \cup \{\bot\}$ is a rather crude model for divergence

- **type-theoretically**, $\bot$ is a **decidable** part of $TX$, hence, non-termination is decidable (!)
- **categorically**, $T$ classifies to few partial functions

This led Capretta to the **delay monad**\(^2\)

$$DX = \nu \gamma. X + \gamma$$

which in ZFC is simply $\mathbb{N} \times X \cup \{\bot\}$

---

\(^2\)Capretta 2005, General recursion via coinductive types
The usage scenario for $\mathbb{D}$ envisages quotienting $DX$ by weak bisimilarity

$$\text{later}(x) \approx x$$

where $[\text{now, latter}] : X + DX \cong DX$ is the inverse of the final coalgebra structure $\text{out} : DX \cong X + DX$

Equivalently, forming coequalizer

$$
\begin{array}{ccc}
DX & \xrightarrow{\text{later}} & DX \\
\downarrow\text{id} & & \downarrow\rho_x \\
DX & \xrightarrow{\rho_x} & D \cong X
\end{array}
$$

yields the relevant quotient $\mathbb{D}_\cong$, the partiality monad

Again, in ZFC $\mathbb{D}_\cong$ collapses back to the maybe monad
Is $\mathcal{D}$ an Elgot monad?

- In ZFC we know that $\mathcal{D}$ is Elgot, because it is order-enriched. But a more abstract reason is that $\rho$ has a right inverse $\nu$ and $(\rho, \nu)$ is an iteration-congruent retraction$^3$)

- In constructive setting $\rho$ cannot have a right inverse, but $\mathcal{D}$ is provably Elgot, assuming some reasonable principles, such as countable choice$^4$)

**Open Problem:** what are the categorical principles ensuring that $D$ extends to an Elgot monad?

---

$^3$Goncharov, Schröder, Rauch, and Piróg 2017, Unifying Guarded and Unguarded Iteration

$^4$Veltri 2017, A Type-Theoretical Study of Nontermination
Duration Monad for Hybrid Semantics
A Hybrid While-Language

We extend our language by adding real numbers $\mathbb{R}$ as a base type and the construct

\[
\Gamma, t : \mathbb{R} \vdash v : A \\
\Gamma, x : A \vdash v \quad w : 2 \\
\Gamma \vdash c \quad x := t \cdot v \& w : A
\]

allowing us to write programs like

\[
\langle u, v \rangle := \langle u, v \rangle := t \cdot \text{ball}(1, 0, t) \& u \geq 0
\]

while true {

\[
\langle u, v \rangle := \langle u, -0.8v \rangle
\]

\[
\langle u, v \rangle := t \cdot \text{ball}(u, v, t) \& u \geq 0
\]
Small-step reduction $\overset{d}{\rightarrow}$ is now indexed by the duration $d$. We reinterpret $\rightarrow$ as $\overset{0}{\rightarrow}$ when appropriate, otherwise override the rules:

\[
\begin{align*}
p \overset{d}{\rightarrow} p' \\
x := p; q \overset{d}{\rightarrow} x := p'; q
\end{align*}
\]

$\forall s \leq d. w[v[s/t]/x] = true$ \hspace{1cm} $\forall e > 0. \exists s \in (d, d + e). w[v[s/t]/x] = false$

\[
\begin{align*}
x := t. v & \& w \overset{d}{\rightarrow} [v[d/t]]
\end{align*}
\]

$\forall s < d. w[v[s/t]/x] = true$ \hspace{1cm} $w[v[d/t]/x] = false$

\[
\begin{align*}
x := t. v & \& w \overset{d}{\rightarrow}
\end{align*}
\]

\[
\begin{align*}
p \overset{d}{\rightarrow} \\
x := p \text{ while } v \{q\} \overset{d}{\rightarrow}
\end{align*}
\]

\[
\begin{align*}
p \overset{d}{\rightarrow} p' \\
x := p \text{ while } v \{q\} \overset{d}{\rightarrow} x := p' \text{ while } v \{q\}
\end{align*}
\]

$w[v/x] = true$

\[
\begin{align*}
x := [v] \text{ while } w \{q\} \overset{0}{\rightarrow} x := q[v/x] \text{ while } w \{q\}
\end{align*}
\]
Big-Step Semantics

- \( p \downarrow d, \nu \) means \( p \) delivers a value \( \nu \) in time \( d \in \mathbb{R}_+ \)
- \( p \downarrow \nu \) means \( p \) diverges in time \( d \in \bar{\mathbb{R}}_+ = \mathbb{R}_+ \cup \{\infty\} \)

Some rules:

\[
\begin{align*}
\frac{p \downarrow d}{x := p; q \downarrow d} & \quad \frac{p \downarrow d, \nu \quad q[\nu/x] \downarrow e}{x := p; q \downarrow d + e} & \quad \frac{p \downarrow d, \nu \quad q[\nu/x] \downarrow e, w}{x := p; q \downarrow d + e, w} \\
[\nu] \downarrow 0, \nu & \quad \frac{p \downarrow d}{x := p \text{ while } \nu \{q\} \downarrow d} & \quad \frac{p \downarrow d, w \quad \nu[w/x] = \text{false}}{x := p \text{ while } \nu \{q\} \downarrow d, w} \\
\end{align*}
\]

\[
\begin{align*}
\frac{p \downarrow d, w \quad \nu[w/x] = \text{true}}{x := q[w/x] \text{ while } \nu \{q\} \downarrow r} & \quad \frac{p \downarrow d, w \quad \nu[w/x] = \text{true}}{x := q[w/x] \text{ while } \nu \{q\} \downarrow r, u} \\
\end{align*}
\]

\[
\begin{align*}
(q_i \downarrow d_i, w_i)_{i \in \omega} \quad \forall i \in \omega. \nu[w_i/x] = \text{true} & \quad q_{i+1} = q[w_i/x] \\
( q_0 \downarrow \sum d_i) & \quad x := q_0 \text{ while } \nu \{q\} \downarrow \sum d_i
\end{align*}
\]
Equivalence of Big-Step and Small-Step

We define \( \rightarrow^d \) as a “transitive closure” of \( \rightarrow^d \).

**Theorem:** \( p \rightarrow^d [v] \) iff \( p \downarrow d, v \) and \( p \rightarrow^d \) iff \( p \downarrow v \)

**Proof Idea:** same same (but different)
Big-step semantics suggests to define the duration monad $\mathcal{Q}$:

$$QX = \mathbb{R}_+ \times X \cup \tilde{\mathbb{R}}_+, \eta(x) = \langle 0, x \rangle, \text{ and}$$

$$\begin{align*}
(f : X \to QY)^*(d, x) &= \langle d + e, y \rangle \quad \text{if } f(x) = \langle e, y \rangle, \\
(f : X \to QY)^*(d, x) &= d + e \quad \text{if } f(x) = e, \\
(f : X \to QY)^*(d) &= d.
\end{align*}$$

**Proposition:** $\mathcal{Q}$ is a monad, as an instance of the generalized writer monad $TX = M \times X + E$ where

- $M$ is a monoid
- $E$ is a monoid module under some $a : M \times E \to E$
Towards Elgotness

\( \mathcal{Q} \) yields neither inductive nor coinductive semantics:

- we cannot order-enrich \( \mathcal{Q} \), for there is no canonical choice of divergence among \( \mathcal{R}_+ \subseteq QX \)
- durations play roles of observables, but the semantics does not distinguish \( p \xrightarrow{d} q \xrightarrow{e} r \) from \( p \xrightarrow{d+e} r \)

We thus define more fine grained layered duration monad

\[ \hat{Q}X = \nu \gamma. (X + R_+ \times \gamma) \], which induces a monad \( \mathcal{Q} \) by a general argument\(^5\)

\(^5\)Uustalu 2003, Generalizing Substitution
\(\hat{Q}\) is an analogue of the delay monad. It also simplifies in ZFC:

\[
\hat{Q}X = \mathbb{R}_+^* \times X \cup \mathbb{R}_+^\omega, \quad \eta(x) = \langle \epsilon, x \rangle \in QX, \text{ and}
\]

\[
(f : X \rightarrow \mathbb{R}_+^* \times Y \cup \mathbb{R}_+^\omega)^*(w, x) = \langle uw, y \rangle \quad \text{if } f(x) = \langle u, y \rangle \in \mathbb{R}_+^* \times Y
\]

\[
(f : X \rightarrow \mathbb{R}_+^* \times Y \cup \mathbb{R}_+^\omega)^*(w, x) = uw \quad \text{if } f(x) = u \in \mathbb{R}_+^\omega
\]

\[
(f : X \rightarrow \mathbb{R}_+^* \times Y \cup \mathbb{R}_+^\omega)^*(w) = w
\]

\(\hat{Q}\) is completely iterative\(^6\), i.e. for every guarded \(f : X \rightarrow \hat{Q}(Y + X)\), there is unique solution \(f^\dagger : X \rightarrow \hat{Q}Y\) of equation \(f^\dagger = [\eta, f^\dagger]^*f\),

where \(f : X \rightarrow \hat{Q}(Y + Z)\) is guarded if it factors as

\[
X \rightarrow (\mathbb{R}_+^* \times Y) \cup (\mathbb{R}_+^* \times Z) \cup \mathbb{R}_+^\omega \leftrightarrow (\mathbb{R}_+^* \times Y) \cup (\mathbb{R}_+^* \times Z) \cup \mathbb{R}_+^\omega \cong Q(Y + Z)
\]

\(^6\)Milius 2005, Completely iterative algebras and completely iterative monads
Let $\hat{Q}_\approx X$ be the quotient of $\hat{Q} X$ under the “weak bisimulation” relation $\approx$ generated by the clauses

$$\langle x, r_1 \ldots r_n \rangle \approx \langle x, s_1 \ldots s_m \rangle, \quad r_1 \ldots \approx s_1 \ldots, \quad \left( \text{where } \sum_i r_i = \sum_j s_j \right)$$

Let $\rho_X : \hat{Q} X \to \hat{Q}_\approx X$ be the emerging quotienting map:

$$\rho_X(x, r_1 \ldots r_n) = \langle x, \sum_i r_i \rangle, \quad \rho_X(r_1 r_2 \ldots) = \sum_i r_i,$$

**Theorem:**

- $\rho$ has a right inverse $\nu : \hat{Q}_\approx \to \hat{Q}$
- $(\rho, \nu)$ is an iteration-congruent retraction, hence $\hat{Q}_\approx$ is Elgot
- $\hat{Q}_\approx \cong Q$
Soundness and Adequacy

We extend the denotational semantics given before with the clause

\[
\llbracket \Gamma, t : R \vdash_v v : A \rrbracket = h
\]
\[
\llbracket \Gamma, x : A \vdash_v w : 2 \rrbracket = b
\]

\[
\lambda \bar{x}. \sup \{ e \mid \forall t \in [0, e]. b(\bar{x}, h(\bar{x}, t)) = \text{true} \} = d
\]
\[
\llbracket \Gamma \vdash_c x := t. v \land w : A \rrbracket =
\]
\[
\lambda \bar{x}. \text{if } b(\bar{x}, h(\bar{x}, d(\bar{x}))) \text{ then } \langle d(\bar{x}), h(\bar{x}, d(\bar{x})) \rangle \text{ else } d(\bar{x})
\]

Theorem (Soundness and Adequacy):

- \( p \downarrow d \) iff \( \llbracket \neg \vdash_c p : A \rrbracket = d \in QA \)
- \( p \downarrow d, v \) iff \( \llbracket \neg \vdash_c p : A \rrbracket = \langle d, \llbracket \neg \vdash_v v : A \rrbracket \rangle \in QA \)
Further Work

- Solve the open problem;
- Make way from duration semantic to evolution semantics (close to finish in work with Renato)
- Transfer the quotienting scenario from Set to Top (seems to work)
- Logics for hybrid programs
- Higher order hybrid semantics (long term goal)


