An Adequate Semantics for Hybrid While

Sergey Goncharov\textsuperscript{a}  Renato Neves\textsuperscript{b}

QAPL/ETAPS 2019, April 6, Prague

\textsuperscript{a}Friedrich-Alexander-Universität Erlangen-Nürnberg
\textsuperscript{b}INESC TEC (HASLab) & University of Minho
Example: Bouncing Ball

Bouncing ball is a simple Newtonian system specified by differential equation $\ddot{h} = -g$ ($g \approx 9.8$) whose solution is

$$h(t) = h_0 + v_0 t - \frac{gt^2}{2}$$

with initial values:

- $v_0 = 0$, $h_0 \neq 0$ (peak height)
- $h_0 = 0$, $v_0 \neq 0$ (zero height)

Features:

- deterministic
- hybrid: the velocity changes discretely at the bottom $v \leftrightarrow -cv$, but it changes continuously in the meanwhile
- Zeno behaviour: the state of rest is only reachable in the limit
Semantics of Hybridness

Stepping stones:

- Hybrid automata [Alur, Courcoubetis, Henzinger, and Ho, 1993], [Henzinger, 1996]
- Platzer’s differential dynamic logic [Platzer, 2008]
- Moggi’s computational monads [Moggi, 1991]
- Kick, Power and Simpson’s evolution comonad [Kick, Power, and Simpson, 2006]
- Hybrid monads (with iteration) [Neves, Barbosa, Hofmann, and Martins, 2016], [Goncharov, Jakob, and Neves, 2018]

Here:

- **HybCore** – a deterministic hybrid call-by-value while-language
- Adequate operational semantics w.r.t. hybrid and duration monads
Prelude: A Simple Adequate While-Language
A simple call-by-value while-language:

- **types** are given with 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 \( f : A \rightarrow B \) with atomic \( B \)

---

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

Values:

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

Computations:

\[ \begin{align*}
\Gamma \vdash_c p : A & \quad \Gamma, x : A \vdash_c q : B \\
\Gamma \vdash_c x := p ; q : B & \\
\Gamma \vdash_v v : A & \\
\Gamma \vdash_c \lfloor \nu \rfloor : A \\
\Gamma \vdash_v v : 2 & \quad \Gamma \vdash_c p : A \quad \Gamma \vdash_c q : A \\
\Gamma \vdash_c \text{if } v \text{ then } p \text{ else } q : A & \\
\Gamma \vdash_c p : A \quad \Gamma, x : A \vdash_v v : 2 \\
\Gamma \vdash_c x := p \text{ while } v \{ q \} : A &
\end{align*} \]
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:

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

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

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

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

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

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

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

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

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

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

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

$$\begin{align*}
p \Downarrow w & \quad v[w/x] = \text{true} \\
x & := q[w/x] \text{ while } v \{ q \} \Downarrow u \\
x & := p \text{ while } v \{ q \} \Downarrow u
\end{align*}$$

Proposition: $p \Downarrow w$ iff $p \rightarrow^{*} [w]$
Definition (Monad)

A monad $\mathbb{T}$ over category $\mathcal{C}$ is given by a Kleisli triple $(T, \eta, -^*)$ where

- $T$ is an endomap on $|\mathcal{C}|$
- $\eta$ is a family of morphisms $\eta_X : X \to TX$, called monad unit
- $(-)^*$ assigns to each $f : X \to TY$ a morphism $f^* : TX \to TY$

satisfying the laws: $\eta^* = \text{id}$, $f^* \eta = f$, $(f^* g)^* = f^* g^*$

This means that the hom-sets $\text{Hom}(X, TY)$ form a category (Kleisli category) under Kleisli composition $f \triangleright g = f^* g$ and $\eta_X \in \text{Hom}(X, TX)$

Definition (Elgot Monad)

$\mathbb{T}$ with an iteration operator $(-)^\dagger : \text{Hom}(X, T(Y + X)) \leftrightarrow \text{Hom}(X, TY)$, satisfying axioms of iteration (omitted) is called Elgot
Denotational Semantics

Assuming $\Gamma = (x_1 : A_1, \ldots, x_n : A_n)$, we interpret

$$\llbracket \Gamma \vdash v : A \rrbracket : A_1 \times \cdots \times A_n \to A$$

$$\llbracket \Gamma \vdash c\ p : A \rrbracket : A_1 \times \cdots \times A_n \to TA$$

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

Moreover, e.g.

$$\llbracket \Gamma \vdash c\ p : A \rrbracket = h$$

$$\llbracket \Gamma, x : A \vdash c\ q : B \rrbracket = u$$

$$\llbracket \Gamma \vdash v : A \rrbracket = h$$

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

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

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

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

using the fact that $\top$ is Elgot

Elgot iteration: $(f : X \to T(Y + X)) \leftrightarrow (f^\dagger : X \to TY)$
Proposition (Soundness): if $p \Downarrow v$ then $\llbracket \vdash_c p : A \rrbracket = v$

Proof Idea: induction over the derivation of $p \Downarrow v$

Proposition (Adequacy): if $\llbracket \vdash_c p : A \rrbracket = v$ then $p \Downarrow v$

Proof Idea: induction over the denotational semantic rules
HybCore: Adding Hybrid Loops
We simply add the type of real numbers $\mathbb{R}$, and the construct

$$
\Gamma, t : \mathbb{R} \vdash \nu : A \quad \Gamma, x : A \vdash \nu \ w : 2 \\
\Gamma \vdash \_ c \ x : t. \nu \ w : A
$$

For example, for the bouncing ball behaviour:

$$
\langle u, v \rangle := (\langle u, v \rangle := t. \text{ball}(1, 0, t) \ & u \geq 0)
$$

while true {

$$
\langle u, v \rangle := \langle u, -0.8v \rangle \\
\langle u, v \rangle := t. \text{ball}(u, v, t) \ & u \geq 0
$$

}
## Types of Hybrid Loops

<table>
<thead>
<tr>
<th></th>
<th>Progressive</th>
<th>Non-progressive</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Convergent</strong></td>
<td>(x := 5) while (x &gt; 0)</td>
<td>(x := 5) while (x &gt; 0)</td>
</tr>
<tr>
<td></td>
<td>({\text{wait}(1); x := x - 1})</td>
<td>({x := x - 1})</td>
</tr>
<tr>
<td><strong>Divergent</strong></td>
<td>(x := 0) while true</td>
<td>(x := 0) while true</td>
</tr>
<tr>
<td></td>
<td>({\text{wait}(1); x := x + 1})</td>
<td>({x := x + 1})</td>
</tr>
<tr>
<td><strong>Zeno</strong></td>
<td>(x := 1) while true</td>
<td></td>
</tr>
<tr>
<td></td>
<td>({\text{wait}(x); x := x/2})</td>
<td></td>
</tr>
</tbody>
</table>
Big-Step Duration Semantics

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

Some selected rules:

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

Zeno behavior

\[
(q_i \downarrow d_i, w_i)_{i \in \omega} \quad \forall i \in \omega. \, \nu[w_i/x] = \text{true} \land q_{i+1} = q[w_i/x] \\
\]

\[
x := q_0 \text{ while } \nu \{q\} \downarrow \sum_i d_i
\]
Big-step semantics suggests to define the duration monad $Q$:

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

and

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

$$(f : X \to QY)^*(d, x) = d + e \quad \text{if} \quad f(x) = e,$$

$$(f : X \to QY)^*(d) = d.$$

**Theorem:** $Q$ is an Elgot monad

Remarkably,

- $Q$ is an instance of the generalized writer monad $TX = M \times X + E$ with a monoid $M$ and an $M$-monoid module $E$
- Elgot iteration operator of $Q$ is neither a least nor unique fixpoint
Since our previous denotational semantics is generic w.r.t. an Elgot monad, we only add a clause for \([\Gamma \vdash_c x := t \cdot v \& w : A]\) (omitted here)

**Theorem (Soundness and Adequacy):**

- \(p \Downarrow d\) iff \([\neg \vdash_c p : A] = d \in QA\)
- \(p \Downarrow d, v\) iff \([\neg \vdash_c p : A] = \langle d, [\neg \vdash_v v : A] \rangle \in QA\)
Big-Step Evolution Semantics

$p, t \Downarrow v$ means: $p$ evaluates to $v$ at time instant $t \in \mathbb{R}_+$

Some selected rules:

\[
\frac{(p, t \Downarrow v_s)_{s \leq t} (q[v_s/x], 0 \Downarrow w_s)_{s \leq t}}{x := p; q, t \Downarrow w_t}
\]

\[
\frac{p \Downarrow d, v', p, d \Downarrow v q[v/x], t \Downarrow w}{x := p; q, d + t \Downarrow w}
\]

\[
\frac{p \Downarrow d}{p \Downarrow d, p, t \Downarrow w \text{ } t < d}
\]

\[
\frac{x := p \text{ while } v \{q\}, t \Downarrow w}{x := p \text{ while } v \{q\}, d \Downarrow w}
\]

\[
\frac{p \Downarrow d, w' p, d \Downarrow w v[w/x] = \text{false}}{x := p \text{ while } v \{q\}, d \Downarrow w}
\]

\[
\frac{p \Downarrow d, w v[w/x] = \text{true}}{x := q[w/x] \text{ while } v \{q\}, t \Downarrow u}
\]

Remarkably, while, is not an iterated sequential composition anymore (!)
The *hybrid monad* $H$ is build on the following functor

$$HX = \{ \langle cc, d, e : [0, d] \to X \rangle \mid d \in \mathbb{R}_+ \} \cup \{ \langle cd, d, e : [0, d] \to X \rangle \mid d \in \mathbb{R}_+ \} \cup \{ \langle od, d, e : [0, d] \to X \rangle \mid d \in \bar{\mathbb{R}}_+ \}$$

**Kleisli lifting** for the simple case $e : [0, d] \to X$ and $e'(x)(0) = x$ for all $x$:

$$(\alpha, d', e')^*(cc, d, e) = (\alpha, d + d', \lambda t. \text{if } t < d \text{ then } e(t) \text{ else } e'(t - d))$$

**Theorem (Soundness and Adequacy):** Given $\vdash_c p : A$,

1. $p, t \Downarrow v$ iff $\llbracket \vdash_c p : A \rrbracket = \langle \alpha, d, e \rangle$, $t < d$ and $e^t = v$
2. $p, d \Downarrow v$ and $p \Downarrow d, w$ iff $v = w$, $\llbracket \vdash_c p : A \rrbracket = \langle cc, d, e \rangle$ and $e^d = v$
Further Work

- Transfer the duration and the hybrid monads from \textbf{Set} to \textbf{Top}
- Program logics for hybrid programs
- Principled combinations of hybridness with other effects (e.g. via monad transformers, colimits, tensors)
- Higher order hybrid semantics and hybrid recursion (long term goal)
Questions?
References


Equivalence of Semantics and Determinacy

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

\[ p \Downarrow w \iff p \rightarrow^* \left[ w \right] \]

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

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

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

**Proposition (Determinacy):** \( p \Downarrow v \) for at most one \( v \)
Small-Step Duration Semantics

Small-step reduction $\rightarrow^d$ is now indexed by durations $d$. We reinterpret $\rightarrow$ as $\rightarrow^0$ when appropriate, otherwise override the rules:

\[
\begin{align*}
\frac{p \rightarrow^d p'}{x := p; q \rightarrow^d x := p'; q} & \quad \frac{p \rightarrow^d p'}{x := p; q \rightarrow^d x := p'; q}
\end{align*}
\]

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

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

$\forall s < d. w[v[s/t]/x] = \text{true} \quad w[v[d/t]/x] = \text{false}$

$\begin{align*}
x := t. v & \quad \& \quad w \rightarrow^d
\end{align*}$

$\begin{align*}
\frac{p \rightarrow^d}{x := p \text{ while } v \{q\} \rightarrow^d} & \quad \frac{p \rightarrow^d p'}{x := p \text{ while } v \{q\} \rightarrow^d x := p' \text{ while } v \{q\}}
\end{align*}$

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

$\begin{align*}
x := [v] \text{ while } w \{q\} \rightarrow^0 x := q[v/x] \text{ while } w \{q\}
\end{align*}$
Equivalence of Big-Step and Small-Step

We define $\implies_{\text{transitive closure}}$ as a “transitive closure” of $\implies_d$.

**Theorem:** $p \implies_{\text{transitive closure}} \llbracket v \rrbracket$ iff $p \Downarrow d$, $v$ and $p \implies_d \llbracket v \rrbracket$ iff $p \Downarrow v$

**Proof Idea:** same same (but different)
Towards Elgotness

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

- we cannot order-enrich \( \mathbb{Q} \), for there is no canonical choice of divergence among \( \bar{R}^- \subseteq QX \)
- 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 \( \mathbb{Q} \) by a general argument\(^2\)

\(^2\)Uustalu 2003, Generalizing Substitution
\( \hat{Q} \) is an analogue of the delay monad. It also simplifies in ZFC:

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

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

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

\[
(f : X \to R^*_+ \times Y \cup R^\omega_+)^*(w) = w
\]

\( \hat{Q} \) is completely iterative\(^3\), i.e. for every guarded \( f : X \to \hat{Q}(Y + X) \), there is unique solution \( f^\dagger : X \to \hat{Q}Y \) of equation \( f^\dagger = [\eta, f^\dagger]^* f \),

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

\[
X \to (R^*_+ \times Y) \cup (R^+_+ \times Z) \cup R^\omega_+ \leftrightarrow (R^*_+ \times Y) \cup (R^*_+ \times Z) \cup R^\omega_+ \cong Q(Y + Z)
\]

\(^3\) 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 $\upsilon : \hat{Q}_\approx \to \hat{Q}$
- $(\rho, \upsilon)$ is an iteration-congruent retraction, hence $\hat{Q}_\approx$ is Elgot
- $\hat{Q}_\approx \cong Q$