Pretty-Big-Step Semantics

Arthur Charguéraud

December 7, 2018

Presentation by Aurèle Barrière

Pretty-Big-Step Semantics

- Operational Semantics.
- Inspired by Big-Step Semantics.

Small-Step Semantics

Step: $(\mathit{code}, \mathit{State}) \rightarrow (\mathit{code}, \mathit{State})$.

Many steps until final configuration.

Big-Step Semantics

One step: $(\mathit{code}, \mathit{State}) \rightarrow \mathit{State}$.

Small-step Reminder

$$\frac{}{(x := a, \sigma) \to \sigma[x \mapsto \mathcal{A}[\![a]\!]\sigma]}\ \text{ASSIG} \qquad \frac{}{(\,\texttt{skip}\,, \sigma) \to \sigma}\ \text{SKIP}$$

$$\frac{(c_1, \sigma) \to \sigma'}{(c_1\ ;\ c_2, \sigma) \to (c_2, \sigma')}\ \text{SEQ1} \qquad \frac{(c_1, \sigma) \to (c_1', \sigma')}{(c_1\ ;\ c_2, \sigma) \to (c_1'\ ;\ c_2, \sigma')}\ \text{SEQ2}$$

$$\frac{}{(\,\texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2, \sigma) \to (c_1, \sigma)}\ \text{IFT} \qquad \textit{if}\ \mathcal{B}[\![b]\!]\,\sigma = \mathbf{tt}$$

$$\frac{}{(\,\texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2, \sigma) \to (c_2, \sigma)}\ \text{IFE} \qquad \textit{if}\ \mathcal{B}[\![b]\!]\,\sigma = \mathbf{ff}$$

$$\frac{}{(\,\texttt{while}\ b\ \texttt{do}\ c, \sigma) \to (\,\texttt{if}\ b\ \texttt{then}\ (c\ ;\ \texttt{while}\ b\ \texttt{do}\ c)\ \texttt{else}\ \texttt{skip}\,, \sigma)}\ \text{WHI}$$

Big-step Reminder

$$\frac{}{(x := a, \sigma) \Downarrow \sigma[x \mapsto \mathcal{A}[\![a]\!]\sigma]} \text{ ASSIG}$$

$$\frac{(S_1, \sigma) \Downarrow \sigma' \qquad (S_2, \sigma') \Downarrow \sigma''}{(S_1 \texttt{ ; } S_2, \sigma) \Downarrow \sigma''} \text{ SEQ}$$

$$\frac{(S_1, \sigma) \Downarrow \sigma'}{(\texttt{ if } b \texttt{ then } S_1 \texttt{ else } S_2, \sigma) \Downarrow \sigma'} \text{ IFT} \qquad \textit{if } \mathcal{B}[\![b]\!]\sigma = \mathbf{tt}$$

$$\frac{(S_2, \sigma) \Downarrow \sigma'}{(\texttt{ if } b \texttt{ then } S_1 \texttt{ else } S_2, \sigma) \Downarrow \sigma'} \text{ IFE} \qquad \textit{if } \mathcal{B}[\![b]\!]\sigma = \mathbf{ff}$$

$$\frac{(S, \sigma) \Downarrow \sigma' \qquad (\texttt{ while } b \texttt{ do } S, \sigma') \Downarrow \sigma''}{(\texttt{ while } b \texttt{ do } S, \sigma) \Downarrow \sigma''} \text{ WHI1} \qquad \textit{if } \mathcal{B}[\![b]\!]\sigma = \mathbf{tt}$$

$$\frac{}{(\texttt{ while } b \texttt{ do } S, \sigma) \Downarrow \sigma} \text{ WHI2} \qquad \textit{if } \mathcal{B}[\![b]\!]\sigma = \mathbf{ff}$$

Why Pretty-Big-Step Semantics?

Big-Step is still used

- 17 out of 40 operational semantics in recent conferences.
- Cost semantics, informal description.
- Some proofs need to be done on big-step semantics.

Choosing a Semantics' style is about the way of writing the rules and doing the proofs, not expressivity.

Drawbacks of Big-Step Semantics

Redundancy when adding new constructs.
For instance: divergence, errors, control-flow exceptions...

Duplicating rules in Big-Step Semantics

The example of divergent behavior and exceptions.

$$\frac{t_1{}_{/m_1} \Rightarrow \mathsf{false}_{/m_2}}{\mathsf{for}\, t_1\, t_2\, t_3{}_{/m_1} \Rightarrow tt_{/m_2}}$$

$$\frac{t_1{}_{/m_1} \Rightarrow \mathsf{true}_{/m_2} \quad t_3{}_{/m_2} \Rightarrow tt_{/m_3} \quad t_2{}_{/m_3} \Rightarrow tt_{/m_4} \quad \mathsf{for}\, t_1\, t_2\, t_3{}_{/m_4} \Rightarrow tt_{/m_5}}{\mathsf{for}\, t_1\, t_2\, t_3{}_{/m_1} \Rightarrow tt_{/m_5}}$$

$$\frac{t_1{}_{/m_1} \Rightarrow^{\mathsf{exn}}{}_{/m_2}}{\mathsf{for}\, t_1\, t_2\, t_3{}_{/m_1} \Rightarrow^{\mathsf{exn}}{}_{/m_2}} \qquad \frac{t_1{}_{/m_1} \Rightarrow^{\infty}}{\mathsf{for}\, t_1\, t_2\, t_3{}_{/m_1} \Rightarrow^{\infty}}\,\mathsf{co}$$

$$\frac{t_1{}_{/m_1} \Rightarrow \mathsf{true}_{/m_2} \quad t_3{}_{/m_2} \Rightarrow^{\mathsf{exn}}{}_{/m_3}}{\mathsf{for}\, t_1\, t_2\, t_3{}_{/m_1} \Rightarrow^{\mathsf{exn}}{}_{/m_3}} \qquad \frac{t_1{}_{/m_1} \Rightarrow \mathsf{true}_{/m_2} \quad t_3{}_{/m_2} \Rightarrow^{\infty}}{\mathsf{for}\, t_1\, t_2\, t_3{}_{/m_1} \Rightarrow^{\infty}}\,\mathsf{co}$$

$$\frac{\begin{array}{c} t_1{}_{/m_1} \Rightarrow \mathsf{true}_{/m_2} \quad t_3{}_{/m_2} \Rightarrow tt_{/m_3} \\ t_2{}_{/m_3} \Rightarrow^{\mathsf{exn}}{}_{/m_4} \end{array}}{\mathsf{for}\, t_1\, t_2\, t_3{}_{/m_1} \Rightarrow^{\mathsf{exn}}{}_{/m_4}} \qquad \frac{\begin{array}{c} t_1{}_{/m_1} \Rightarrow \mathsf{true}_{/m_2} \quad t_3{}_{/m_2} \Rightarrow tt_{/m_3} \\ t_2{}_{/m_3} \Rightarrow^{\infty} \end{array}}{\mathsf{for}\, t_1\, t_2\, t_3{}_{/m_1} \Rightarrow^{\infty}}\,\mathsf{co}$$

$$\frac{\begin{array}{c} t_1{}_{/m_1} \Rightarrow \mathsf{true}_{/m_2} \quad t_3{}_{/m_2} \Rightarrow tt_{/m_3} \\ t_2{}_{/m_3} \Rightarrow tt_{/m_4} \quad \mathsf{for}\, t_1\, t_2\, t_3{}_{/m_4} \Rightarrow^{\mathsf{exn}}{}_{/m_5} \end{array}}{\mathsf{for}\, t_1\, t_2\, t_3{}_{/m_1} \Rightarrow^{\mathsf{exn}}{}_{/m_5}} \qquad \frac{\begin{array}{c} t_1{}_{/m_1} \Rightarrow \mathsf{true}_{/m_2} \quad t_3{}_{/m_2} \Rightarrow tt_{/m_3} \\ t_2{}_{/m_3} \Rightarrow tt_{/m_4} \quad \mathsf{for}\, t_1\, t_2\, t_3{}_{/m_4} \Rightarrow^{\infty} \end{array}}{\mathsf{for}\, t_1\, t_2\, t_3{}_{/m_1} \Rightarrow^{\infty}}\,\mathsf{co}$$

Pretty-Big Step Presentation

Properties

- less rules.
- no duplication of premises.
- use coinduction for diverging behaviors.
- add intermediate terms.

Example: λ-calculus

$$v \ := \ \mathsf{int}\, n \ \mid \ \mathsf{abs}\, x\, t \qquad\qquad t \ := \ \mathsf{val}\, v \ \mid \ \mathsf{var}\, x \ \mid \ \mathsf{app}\, t\, t$$

Evaluating Application

Big-Step

$$\frac{t_1 \Rightarrow \mathsf{abs}\,x\,t \qquad t_2 \Rightarrow v \qquad [x \to v]\,t \Rightarrow v'}{\mathsf{app}\,t_1\,t_2 \Rightarrow v'}$$

First Attempt at Pretty-Big-Step

$$\frac{t_1 \Rightarrow v_1 \quad \mathsf{app}\,v_1\,t_2 \Rightarrow v'}{\mathsf{app}\,t_1\,t_2 \Rightarrow v'} \qquad \frac{t_2 \Rightarrow v_2 \quad \mathsf{app}\,v_1\,v_2 \Rightarrow v'}{\mathsf{app}\,v_1\,t_2 \Rightarrow v'} \qquad \frac{[x \to v]\,t \Rightarrow v'}{\mathsf{app}\,(\mathsf{abs}\,x\,t)\,v \Rightarrow v'}$$

Overlapping Problem

Evaluation isn't syntax-directed.

Intermediate Terms

Term Syntax

$$e := \mathsf{trm}\, t \mid \mathsf{app1}\, v\, t \mid \mathsf{app2}\, v\, v$$

Non-overlapping Semantics

$$\frac{}{v \Downarrow v} \qquad \frac{t_1 \Downarrow v_1 \qquad \mathsf{app1}\, v_1\, t_2 \Downarrow v'}{\mathsf{app}\, t_1\, t_2 \Downarrow v'}$$

$$\frac{t_2 \Downarrow v_2 \qquad \mathsf{app2}\, v_1\, v_2 \Downarrow v'}{\mathsf{app1}\, v_1\, t_2 \Downarrow v'} \qquad \frac{[x \to v]\, t \Downarrow v'}{\mathsf{app2}\, (\mathsf{abs}\, x\, t)\, v \Downarrow v'}$$

Example

Evaluation of $(\lambda y.\ (y\ 0))\ (\lambda x.\ x + 2)$.
app (**abs** y (**app** y 0)) (**abs** x $(x + 2)$)).

Generalized Behavior and Extended Intermediate Terms

$b := \mathsf{ret}\, v \mid \mathsf{exn}\, v$

$e := \mathsf{trm}\, t \mid \mathsf{app1}\, b\, t \mid \mathsf{app2}\, v\, b \mid \mathsf{raise1}\, b \mid \mathsf{try1}\, b\, t$

Pretty-Big-Step Rules

$$\frac{}{v \Downarrow v} \qquad \frac{t_1 \Downarrow b_1 \qquad \mathsf{app1}\, b_1\, t_2 \Downarrow b}{\mathsf{app}\, t_1\, t_2 \Downarrow b} \qquad \frac{}{\mathsf{app1}\, (\mathsf{exn}\, v)\, t \Downarrow \mathsf{exn}\, v}$$

$$\frac{t_2 \Downarrow b_2 \qquad \mathsf{app2}\, v_1\, b_2 \Downarrow b}{\mathsf{app1}\, v_1\, t_2 \Downarrow b} \qquad \frac{}{\mathsf{app2}\, v\, (\mathsf{exn}\, v) \Downarrow \mathsf{exn}\, v} \qquad \frac{[x \to v]\, t \Downarrow b}{\mathsf{app2}\, (\mathsf{abs}\, x\, t)\, v \Downarrow b}$$

$$\frac{t \Downarrow b_1 \qquad \mathsf{raise1}\, b_1 \Downarrow b}{\mathsf{raise}\, t \Downarrow b} \qquad \frac{}{\mathsf{raise1}\, v \Downarrow \mathsf{exn}\, v} \qquad \frac{}{\mathsf{raise1}\, (\mathsf{exn}\, v) \Downarrow \mathsf{exn}\, v}$$

$$\frac{t_1 \Downarrow b_1 \qquad \mathsf{try1}\, b_1\, t_2 \Downarrow b}{\mathsf{try}\, t_1\, t_2 \Downarrow b} \qquad \frac{}{\mathsf{try1}\, v\, t \Downarrow v} \qquad \frac{\mathsf{app}\, t\, v \Downarrow b}{\mathsf{try1}\, (\mathsf{exn}\, v)\, t \Downarrow b}$$

Divergence

$$b := \mathsf{ret}\, v \mid \mathsf{exn}\, v \qquad o := \mathsf{ter}\, b \mid \mathsf{div} \qquad e := \mathsf{trm}\, t \mid \mathsf{app1}\, o\, t \mid \mathsf{app2}\, v\, o \mid \mathsf{raise1}\, o \mid \mathsf{try1}\, o\, t$$

$$\frac{}{\mathsf{abort}\,(\mathsf{exn}\, v)} \qquad \frac{}{\mathsf{abort}\,\mathsf{div}} \qquad \frac{}{v \Downarrow v} \qquad \frac{t_1 \Downarrow o_1 \qquad \mathsf{app1}\, o_1\, t_2 \Downarrow o}{\mathsf{app}\, t_1\, t_2 \Downarrow o}$$

$$\frac{\mathsf{abort}\, o}{\mathsf{app1}\, o\, t \Downarrow o} \qquad \frac{t_2 \Downarrow o_2 \qquad \mathsf{app2}\, v_1\, o_2 \Downarrow o}{\mathsf{app1}\, v_1\, t_2 \Downarrow o} \qquad \frac{\mathsf{abort}\, o}{\mathsf{app2}\, v\, o \Downarrow o} \qquad \frac{[x \to v]\, t \Downarrow o}{\mathsf{app2}\,(\mathsf{abs}\, x\, t)\, v \Downarrow o}$$

$$\frac{t \Downarrow o_1 \qquad \mathsf{raise1}\, o_1 \Downarrow o}{\mathsf{raise}\, t \Downarrow o} \qquad \frac{\mathsf{abort}\, o}{\mathsf{raise1}\, o \Downarrow o} \qquad \frac{}{\mathsf{raise1}\, v \Downarrow \mathsf{exn}\, v} \qquad \frac{t_1 \Downarrow o_1 \qquad \mathsf{try1}\, o_1\, t_2 \Downarrow o}{\mathsf{try}\, t_1\, t_2 \Downarrow o}$$

$$\frac{}{\mathsf{try1}\, v\, t \Downarrow v} \qquad \frac{\mathsf{app}\, t\, v \Downarrow o}{\mathsf{try1}\,(\mathsf{exn}\, v)\, t \Downarrow o} \qquad \frac{\mathsf{abort}\, o \qquad \forall v.\ o \neq \mathsf{exn}\, v}{\mathsf{try1}\, o\, t \Downarrow o}$$

Coinduction

```
Section stream.
  Variable A : Type.

CoInductive stream : Type :=
  | Cons : A → stream → stream.
End stream.
```

Guardedness Condition

Every co-recursive call must be guarded by a constructor.

http://adam.chlipala.net/cpdt/html/Coinductive.html

Properties

Lemma 1. *For any term t,* $\quad t \Downarrow \mathsf{div} \;\rightarrow\; \mathsf{False}.$

Lemma 2. *For any term e and outcome o,* $\quad e \Downarrow o \;\rightarrow\; e \Downarrow^{co} o.$

Lemma 3. *For any term e and outcome o,* $\quad e \Downarrow^{co} o \;\rightarrow\; e \Downarrow o \;\vee\; e \Downarrow^{co} \mathsf{div}.$

Theorem 1 (Equivalence with big-step semantics). *For any term t, and for any behavior b (describing either a value or an exception),*

$$t \Downarrow b \text{ if and only if } t \Rightarrow b \qquad \text{and} \qquad t \Downarrow^{co} \mathsf{div} \text{ if and only if } t \Rightarrow^{\infty}.$$

Lemma 4 (Determinacy). $\quad \forall e o_1 o_2.\; e \Downarrow o_1 \;\wedge\; e \Downarrow^{co} o_2 \;\rightarrow\; o_1 = o_2$

Errors and Tying Soundness

Generic Error Rule

$$\frac{\neg\ (e\downarrow)}{e \Downarrow \mathsf{err}}$$

Type Soundness

Theorem 2 (Type soundness). *For any* t *and* T, $\vdash t : T \ \rightarrow\ \neg\ t \Downarrow$ *err*.

Traces

$$\frac{}{\mathsf{abort}\,(\mathsf{ter}\,\tau\,(\mathsf{exn}\,v))} \qquad \frac{}{\mathsf{abort}\,(\mathsf{div}\,\sigma)}$$

$$\frac{}{v \Downarrow \mathsf{ter}\,[\epsilon]\,v} \qquad \frac{t_1 \Downarrow o_1 \qquad \mathsf{app1}\,o_1\,t_2 \Downarrow o}{\mathsf{app}\,t_1\,t_2 \Downarrow [\epsilon]\cdot o} \qquad \frac{\mathsf{abort}\,o}{\mathsf{app1}\,o\,t \Downarrow [\epsilon]\cdot o}$$

$$\frac{t_2 \Downarrow o_2 \qquad \mathsf{app2}\,v_1\,o_2 \Downarrow o}{\mathsf{app1}\,(\mathsf{ter}\,\tau\,v_1)\,t_2 \Downarrow [\epsilon]\cdot\tau\cdot o} \qquad \frac{\mathsf{abort}\,o}{\mathsf{app2}\,v\,o \Downarrow [\epsilon]\cdot o} \qquad \frac{[x \to v]\,t \Downarrow o}{\mathsf{app2}\,(\mathsf{abs}\,x\,t)\,(\mathsf{ter}\,\tau\,v) \Downarrow [\epsilon]\cdot\tau\cdot o}$$

$$\frac{t \Downarrow o_1 \qquad \mathsf{read1}\,o_1 \Downarrow o}{\mathsf{read}\,t \Downarrow [\epsilon]\cdot o} \qquad \frac{\mathsf{abort}\,o}{\mathsf{read1}\,o \Downarrow [\epsilon]\cdot o} \qquad \frac{}{\mathsf{read1}\,(\mathsf{ter}\,\tau\,tt) \Downarrow \mathsf{ter}\,([\epsilon]\cdot\tau\cdot[\mathsf{in}\,n])\,n}$$

$$\frac{t \Downarrow o_1 \qquad \mathsf{write1}\,o_1 \Downarrow o}{\mathsf{write}\,t \Downarrow [\epsilon]\cdot o} \qquad \frac{\mathsf{abort}\,o}{\mathsf{write1}\,o \Downarrow [\epsilon]\cdot o} \qquad \frac{}{\mathsf{write1}\,(\mathsf{ter}\,\tau\,n) \Downarrow \mathsf{ter}\,([\epsilon]\cdot\tau\cdot[\mathsf{out}\,n])\,tt}$$

Lemma 5. *For any finite trace* τ, $\quad (e \Downarrow^{co} \mathit{ter}\,\tau\,v) \;\Leftrightarrow\; (e \Downarrow \mathit{ter}\,\tau\,v)$.

Scaling Up

Side-effects

$$\frac{t_1 \;{}_{/m} \Downarrow o_1 \qquad \mathsf{app1}\, o_1\, t_2 \;{}_{/m} \Downarrow o}{\mathsf{app}\, t_1\, t_2 \;{}_{/m} \Downarrow o} \qquad\qquad \frac{t_2 \;{}_{/m} \Downarrow o_2 \qquad \mathsf{app2}\, v_1\, o_2 \;{}_{/m} \Downarrow o}{\mathsf{app1}\, (\mathsf{ter}\, m\, v_1)\, t_2 \;{}_{/m'} \Downarrow o}$$

Other Functionalities

- For Loops
- Tuples
- Generic Abort Rule

Formalization of core-Caml

Features

- Booleans, Integers, Tuples, Algebraic Data Types, records
- Functions, Recursive functions, applications, sequences
- Conditionals, for loops, while loops
- Pattern-matching, let-bindings, assertions

Missing Features

- Floats
- Mutual Recursion
- With construct for records
- Arrays
- Objects, Modules

Formalization of core-Caml

	rules	premises	tokens
Big-step without divergence	71	83	1540
Big-step with divergence	113	143	2263
Pretty-big-step	70	60	1361

```
(** Grammar of outcomes *)
Inductive out :=
  | out_beh : mem -> beh -> out
  | out_div : out.

(** Grammar of extended terms *)
Inductive ext : Type :=
  | ext_trm : trm -> ext
  | ext_binary_1 : prim -> out -> trm -> ext
  | ext_binary_2 : prim -> val -> out -> ext
  | ext_app_1 : out -> trm -> ext
  ...

Lemma soundness : forall t T,
  typing empty t T -> ~ red t out_err.
```

http://www.chargueraud.org/research/2012/pretty/

Overview

European Symposium of Programming, 2013. 43 citations.

Further Works

- A trusted mechanized JavaScript specification, 2014
- Certified Abstract Interpretation with Pretty-Big-Step Semantics, 2015
- Functional Big-Step Semantics, 2016

Questions

- Determinacy lemma?
- What's specific to Pretty-Big-Step?
- Typing Complexity?
- Standard notion of Coinduction.

Conclusion

- New operational Semantics.
- Inspired by and Equivalent to Big-Step.
- Less rules, more factorization.
- Adding functionalities is easier.
- Implemented a full language (core-Ocaml).
- Size reduction of 40%.
- Co-Inductive proofs cannot be done in Coq yet.