20 Definitions for Hoare Triples

Includes: Big-step Semantics for Non-Deterministic Languages

Arthur Charguéraud

Inria & Université de Strasbourg, CNRS, ICube

January 11th, 2021

Definition of triples

Consider a programming language with a given semantics.

How to define the meaning of a triple w.r.t. the semantics?

$$\{H\}\ t\ \{Q\}$$

Definition of triples

Consider a programming language with a given semantics.

How to define the meaning of a triple w.r.t. the semantics?

$$\{H\}\ t\ \{Q\}$$

For any state s satisfying H, all possible results of t/s satisfy Q.

Definition of triples

Consider a programming language with a given semantics.

How to define the meaning of a triple w.r.t. the semantics?

$$\{H\}\ t\ \{Q\}$$

For any state s satisfying H, all possible results of t/s satisfy Q.

For example, total-correctness triple in deterministic big-step semantics:

$$\{H\}\ t\ \{Q\} \quad \equiv \quad \forall s.\ H\ s \ \Rightarrow\ \exists v s'.\ (t/s \Downarrow v/s') \ \wedge\ Q\ v\ s'$$

Many possible triples

Parameters:

- small-step, or big-step
- deterministic, or non-deterministic
- with stuck terms, or completed with error propagation
- total correctness, or partial correctness, or just divergence
- Hoare Logic, or Separation Logic

Small-step vs big-step

Small-step: $t/s \longrightarrow t'/s'$

Big-step: $t/s \Downarrow v/s'$

Small-step vs big-step

Small-step: $t/s \longrightarrow t'/s'$

Big-step: $t/s \Downarrow v/s'$

Coinductive big-step: $t/s \Uparrow^{\mathsf{co}}$

$$\frac{t_1/s \Uparrow^{\mathsf{co}}}{(\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2)/s \Uparrow^{\mathsf{co}}} \qquad \frac{t_1/s \Downarrow v_1/s' \qquad ([x \to v_1]\, t_2)/s' \Uparrow^{\mathsf{co}}}{(\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2)/s \Uparrow^{\mathsf{co}}}$$

Determinism

Deterministic semantics: at most one transition / at most one result.

Determinism

Deterministic semantics: at most one transition / at most one result.

Non-deterministic semantics: finite or infinite branching. Examples:

$$\textsc{rand-int}\ \frac{0 \leqslant m \leqslant n}{(\mathsf{rand}\, n)/s \longrightarrow m/s} \qquad \textsc{rand-string}\ \frac{w \textit{ a string of any length}}{(\mathsf{rand\_string}\, tt)/s \longrightarrow w/s}$$

Stuck configurations and complete semantics

Stuck configuration: not already a value and no possible transitions.

Stuck configurations and complete semantics

Stuck configuration: not already a value and no possible transitions.

Complete semantics: no stuck configurations.

Stuck configurations and complete semantics

Stuck configuration: not already a value and no possible transitions.

Complete semantics: no stuck configurations.

Any semantics can be completed with error propagation rules, e.g.:

$$\textsc{rand-err}\ \frac{n \leqslant 0}{(\mathsf{rand}\, n)/s \longrightarrow \mathsf{err}/s} \qquad \textsc{let-err}\ \frac{}{(\mathsf{let}\, x = \mathsf{err}\, \mathsf{in}\, t_2)/s \longrightarrow \mathsf{err}/s}$$

Partial and total correctness

Consider a triple $\{H\}\ t\ \{Q\}$ and a state s satisfying H.

Safety: none of the possible evaluation of t/s can get stuck.

Correctness: if t/s can evaluate to v/s', then $Q\,v\,s'$ holds.

Partial and total correctness

Consider a triple $\{H\}\ t\ \{Q\}$ and a state s satisfying H.

Safety: none of the possible evaluation of t/s can get stuck.

Correctness: if t/s can evaluate to v/s', then $Q\,v\,s'$ holds.

Termination: all possible evaluations of t/s are finite.

Divergence: all possible evaluations of t/s are infinite.

Partial and total correctness

Consider a triple $\{H\}\ t\ \{Q\}$ and a state s satisfying H.

Safety: none of the possible evaluation of t/s can get stuck.

Correctness: if t/s can evaluate to v/s', then $Q\,v\,s'$ holds.

Termination: all possible evaluations of t/s are finite.

Divergence: all possible evaluations of t/s are infinite.

Partial correctness: safety + correctness.

Total correctness: safety + correctness + termination.

Separation Logic from Hoare Logic

Separation Logic triples can be defined in terms of Hoare Logic triples.

(*baked-in frame rule*)

$$^{\mathsf{SEP}}\,\{H\}\,t\,\{Q\} \qquad \equiv \qquad \forall H'.\ \{H \star H'\}\,t\,\{Q \star H'\}$$

Thus, let's focus on Hoare triples.

Evaluation predicate

General pattern for triples:

$$\{H\}\ t\ \{Q\} \quad \equiv \quad \forall s.\ H\ s \ \Rightarrow\ \texttt{eval}\ t\ s\ Q$$

where "$\texttt{eval}\ t\ s\ Q$" asserts that any possible result of t/s satisfies Q.

Naming conventions

Various definitions of $\texttt{eval}\, t\, s\, Q$ **are named:**

$$\texttt{evalXYZ}\, t\, s\, Q$$

X: total correctness (**t**), or partial correctness (**p**)

Y: non-deterministic (**n**), or deterministic (**d**), or complete (**c**)

Z: big-step (**b**), or small-step (**s**), (co-)inductive small-step (**z**)

Naming conventions

Various definitions of $\texttt{eval}\, t\, s\, Q$ **are named:**

$$\texttt{evalXYZ}\, t\, s\, Q$$

X: total correctness (**t**), or partial correctness (**p**)

Y: non-deterministic (**n**), or deterministic (**d**), or complete (**c**)

Z: big-step (**b**), or small-step (**s**), (co-)inductive small-step (**z**)

Various definitions of divergence are named:

$$\texttt{divYZ}\, t\, s$$

Let's fill the table

		det.(**d**)	non-det.(**n**)	complete(**c**)
total(**t**)	big (**b**)			
	small(**s**)			
partial(**p**)	big (**b**)			
	small(**s**)			
diverge(**d**)	big (**b**)			
	small(**s**)			

Let's fill the table, step by step

		det.(**d**)	non-det.(**n**)	complete(**c**)
total(**t**)	big (**b**)	3		
	small(**s**)	3		
partial(**p**)	big (**b**)	3		2
	small(**s**)	3	1	2
diverge(**d**)	big (**b**)	3		
	small(**s**)	3	1	

Let's fill the table, step by step

		det.(**d**)	non-det.(**n**)	complete(**c**)
total(**t**)	big (**b**)	3		
	small(**s**)	3	4	4
partial(**p**)	big (**b**)	3		2
	small(**s**)	3	1	2
diverge(**d**)	big (**b**)	3		
	small(**s**)	3	1	

Let's fill the table, step by step

		det.(**d**)	non-det.(**n**)	complete(**c**)
total(**t**)	big (**b**)	3	5	
	small(**s**)	3	4	4
partial(**p**)	big (**b**)	3	5	2
	small(**s**)	3	1	2
diverge(**d**)	big (**b**)	3	5	
	small(**s**)	3	1	

Partial correctness in small-step

Recall: "$\texttt{eval}\, t\, s\, Q$" asserts that any possible output of t/s satisfies Q.

Partial correctness, small-step:
an execution prefix either reaches a valid result, or can make progress.

$$\texttt{evalnps}\, t\, s\, Q \quad \equiv \quad \forall s' t'.\ (t/s \longrightarrow^* t'/s') \Rightarrow \quad \big(\exists v.\ t' = \mathsf{val}\, v\ \wedge\ Q\, v\, s'\big) \vee \big(\mathsf{canprogress}\, t'\, s'\big)$$

where

$$\mathsf{canprogress}\, t\, s \quad \equiv \quad \exists t' s'.\ t/s \longrightarrow t'/s'$$

Divergence in small-step

Divergence $=$ partial correctness with empty postcondition $(\lambda vs.\,\mathsf{False})$

Divergence in small-step

Divergence = partial correctness with empty postcondition $(\lambda vs.\,\mathsf{False})$

$$\texttt{evalnps}\,t\,s\,Q \quad\equiv\quad \forall s't'.\ (t/s \longrightarrow^* t'/s') \Rightarrow \quad \begin{array}{l}(\exists v.\ t' = \mathsf{val}\,v\ \wedge\ Q\,v\,s') \\ \vee\,(\mathsf{canprogress}\,t'\,s')\end{array}$$

Divergence in small-step: any execution prefix can make progress.

$$\texttt{divns}\,t\,s \quad\equiv\quad \forall s't'.\ (t/s \longrightarrow^* t'/s') \Rightarrow \mathsf{canprogress}\,t'\,s'$$

Partial correctness for complete semantics

Complete semantics = no stuck terms.

Partial correctness for complete semantics

Complete semantics = no stuck terms.

$$\texttt{evalnps}\, t\, s\, Q \quad \equiv \quad \forall s' t'.\ (t/s \longrightarrow^* t'/s') \Rightarrow \quad \begin{array}{l} \big(\exists v.\ t' = \mathsf{val}\, v\ \wedge\ Q\, v\, s'\big) \\ \vee \big(\mathsf{canprogress}\, t'\, s'\big) \end{array}$$

Partial correctness for complete semantics, small-step:
if an execution terminates, it reaches a valid result, excluding errors.

$$\texttt{evalcps}\, t\, s\, Q \quad \equiv \quad \forall s' v.\ (t/s \longrightarrow^* v/s') \ \Rightarrow\ v \neq \mathsf{err}\ \wedge\ Q\, v\, s'$$

Partial correctness for complete semantics

Complete semantics = no stuck terms.

$$\texttt{evalnps}\, t\, s\, Q \quad \equiv \quad \forall s' t'.\ (t/s \longrightarrow^* t'/s') \Rightarrow \quad \begin{array}{l} \big(\exists v.\ t' = \mathsf{val}\, v\ \wedge\ Q\, v\, s'\big) \\ \vee \big(\mathsf{canprogress}\, t'\, s'\big) \end{array}$$

Partial correctness for complete semantics, small-step:
if an execution terminates, it reaches a valid result, excluding errors.

$$\texttt{evalcps}\, t\, s\, Q \quad \equiv \quad \forall s' v.\ (t/s \longrightarrow^* v/s') \ \Rightarrow\ v \neq \mathsf{err}\ \wedge\ Q\, v\, s'$$

Partial correctness for complete semantics, big-step:

$$\texttt{evalcpb}\, t\, s\, Q \quad \equiv \quad \forall s' v.\ (t/s \Downarrow v/s') \ \Rightarrow\ v \neq \mathsf{err}\ \wedge\ Q\, v\, s'$$

Definitions for deterministic semantics

Total correctness for deterministic semantics, big-step:

$$\texttt{evaldtb}\, t\, s\, Q \quad \equiv \quad \exists v s'.\ (t/s \Downarrow v/s') \ \wedge\ Q\, v\, s'$$

Total correctness for deterministic semantics, small-step:

$$\texttt{evaldts}\, t\, s\, Q \quad \equiv \quad \exists v s'.\ (t/s \longrightarrow^* v/s') \ \wedge\ Q\, v\, s'$$

Definitions for deterministic semantics

Total correctness for deterministic semantics, big-step:

$$\texttt{evaldtb}\, t\, s\, Q \quad \equiv \quad \exists v s'.\ (t/s \Downarrow v/s') \ \wedge\ Q\, v\, s'$$

Total correctness for deterministic semantics, small-step:

$$\texttt{evaldts}\, t\, s\, Q \quad \equiv \quad \exists v s'.\ (t/s \longrightarrow^{*} v/s') \ \wedge\ Q\, v\, s'$$

Partial correctness for deterministic semantics, big-step:
partial correctness $=$ total correctness $\vee$ divergence.

$$\texttt{evaldpb}\, t\, s\, Q \quad \equiv \quad \texttt{evaldtb}\, t\, s\, Q \ \vee\ \texttt{divdb}\, t\, s$$

where $\texttt{divdb}\, t\, s \ \equiv\ (t/s \Uparrow^{\mathsf{co}})$, i.e. coinductive big-step divergence.

What we have so far

		det.(**d**)	non-det.(**n**)	complete(**c**)
total(**t**)	big (**b**)	`evaldtb`		
	small(**s**)	`evaldts`		
partial(**p**)	big (**b**)	`evaldpb`		`evalcpb`
	small(**s**)	`(evalnps)`	`evalnps`	`evalcps`
diverge(**d**)	big (**b**)	`divdb`		
	small(**s**)	`(divns)`	`divns`	`(divns)`

In parenthesis: no simpler definition than the non-deterministic one.

What we have so far

		det.(**d**)	non-det.(**n**)	complete(**c**)
total(**t**)	big (**b**)	`evaldtb`	5	
	small(**s**)	`evaldts`	4	4
partial(**p**)	big (**b**)	`evaldpb`	5	`evalcpb`
	small(**s**)	`(evalnps)`	`evalnps`	`evalcps`
diverge(**d**)	big (**b**)	`divdb`	5	
	small(**s**)	`(divns)`	`divns`	`(divns)`

In parenthesis: no simpler definition than the non-deterministic one.

4. Termination for non-deterministic semantics in small-step?
5. Non-deterministic semantics in big-step?

4. Termination for non-deterministic semantics in small-step

Termination for non-deterministic semantics (1/2)

Idea: termination = bounded execution length.

Termination for non-deterministic semantics:

$$\text{terminates}\, s\, t \quad \equiv \quad \exists N.\ \forall n t' s'.\ (t/s \longrightarrow^n t'/s') \Rightarrow n \leqslant N$$

Termination for non-deterministic semantics (1/2)

Idea: termination = bounded execution length.

Termination for non-deterministic semantics: finite branching only

$$\text{terminates}\, s\, t \quad \equiv \quad \exists N.\ \forall n t' s'.\ (t/s \longrightarrow^n t'/s') \Rightarrow n \leqslant N$$

Too restrictive for infinitely-branching non-determinism. Example:

```
String.iter (fun c => ()) (rand_string())
```

Termination for non-deterministic semantics (2/2)

Conjecture: cannot capture termination in general using $\longrightarrow^*$ or $\longrightarrow^n$.

Termination for non-deterministic semantics (2/2)

Conjecture: cannot capture termination in general using $\longrightarrow^*$ or $\longrightarrow^n$.

Termination for non-deterministic semantics, inductively:

$$\frac{}{\text{terminates}\, v\ s} \qquad \frac{\forall t'\, s'.\ (t/s \longrightarrow t'/s') \ \Rightarrow\ \text{terminates}\, t'\ s'}{\text{terminates}\, t\ s}$$

Termination for non-deterministic semantics (2/2)

Conjecture: cannot capture termination in general using $\longrightarrow^*$ or $\longrightarrow^n$.

Termination for non-deterministic semantics, inductively:

$$\frac{}{\mathsf{terminates}\, v\, s} \qquad \frac{\forall t'\, s'.\ (t/s \longrightarrow t'/s') \ \Rightarrow\ \mathsf{terminates}\, t'\, s'}{\mathsf{terminates}\, t\, s}$$

Total correctness in small-step, as partial correctness + termination:

$$\texttt{evalnts}\, t\, s\, Q \quad \equiv \quad \texttt{evalnps}\, t\, s\, Q \ \wedge\ \mathsf{terminates}\, t\, s$$

Termination for non-deterministic semantics (2/2)

Conjecture: cannot capture termination in general using $\longrightarrow^*$ or $\longrightarrow^n$.

Termination for non-deterministic semantics, inductively:

$$\frac{}{\mathsf{terminates}\, v\, s} \qquad \frac{\forall t' s'.\ (t/s \longrightarrow t'/s') \ \Rightarrow\ \mathsf{terminates}\, t'\, s'}{\mathsf{terminates}\, t\, s}$$

Total correctness in small-step, as partial correctness + termination:

$$\mathtt{evalnts}\, t\, s\, Q \quad \equiv \quad \mathtt{evalnps}\, t\, s\, Q \ \wedge\ \mathsf{terminates}\, t\, s$$

$$\mathtt{evalnps}\, t\, s\, Q \ \equiv\ \forall s' t'.\ (t/s \longrightarrow^* t'/s') \Rightarrow \quad \big(\exists v.\ t' = \mathsf{val}\, v\ \wedge\ Q\, v\, s'\big) \vee \big(\mathsf{canprogress}\, t'\, s'\big)$$

Total correctness in small-step, inductively

$\texttt{evalntz}\, t\, s\, Q$: any evaluation of t/s terminates on a result satisfying Q.

Total correctness in small-step, inductively:

$$\frac{Q\, v\, s}{\texttt{evalntz}\, v\, s\, Q} \qquad \frac{\mathsf{canprogress}\, t\, s \quad \left(\forall t' s'.\ \ t/s \longrightarrow t'/s' \ \Rightarrow\ \texttt{evalntz}\, t'\, s'\, Q\right)}{\texttt{evalntz}\, t\, s\, Q}$$

Total correctness in small-step, inductively

$\texttt{evalntz}\, t\, s\, Q$: any evaluation of t/s terminates on a result satisfying Q.

Total correctness in small-step, inductively:

$$\frac{Q\, v\, s}{\texttt{evalntz}\, v\, s\, Q} \qquad \frac{\mathsf{canprogress}\, t\, s \quad \big(\forall t' s'.\ \ t/s \longrightarrow t'/s' \ \Rightarrow\ \texttt{evalntz}\, t'\, s'\, Q\big)}{\texttt{evalntz}\, t\, s\, Q}$$

Specialization for complete semantics:
replace "$\mathsf{canprogress}\, t\, s$" with "$\mathsf{notavalue}\, t$", defined as $\forall v.\ t \neq \mathsf{val}\, v$.

Total correctness in small-step, inductively

$\texttt{evalntz}\, t\, s\, Q$: any evaluation of t/s terminates on a result satisfying Q.

Total correctness in small-step, inductively:

$$\frac{Q\, v\, s}{\texttt{evalntz}\, v\, s\, Q} \qquad \frac{\texttt{canprogress}\, t\, s \quad \left(\forall t' s'.\ t/s \longrightarrow t'/s' \ \Rightarrow\ \texttt{evalntz}\, t'\, s'\, Q\right)}{\texttt{evalntz}\, t\, s\, Q}$$

Specialization for complete semantics:
replace "canprogress $t\, s$" with "notavalue t", defined as $\forall v.\ t \neq \mathsf{val}\, v$.

Specialization for deterministic semantics:

$$\frac{Q\, v\, s}{\texttt{evaldtz}\, v\, s\, Q} \qquad \frac{t/s \longrightarrow t'/s' \qquad \texttt{evaldtz}\, t'\, s'\, Q}{\texttt{evaldtz}\, t\, s\, Q}$$

Coinductive partial correctness in small-step

Partial correctness for non-deterministic semantics, coinductively:

$$\frac{Q\,v\,s}{\texttt{evalnpz}\,v\,s\,Q} \qquad \frac{\textsf{canprogress}\,t\,s \quad \left(\forall t' s'.\ \ t/s \longrightarrow t'/s' \ \Rightarrow\ \texttt{evalnpz}\,t'\,s'\,Q\right)}{\texttt{evalnpz}\,t\,s\,Q}$$

Coinductive partial correctness in small-step

Partial correctness for non-deterministic semantics, coinductively:

$$\frac{Q\,v\,s}{\texttt{evalnpz}\,v\,s\,Q} \qquad \frac{\mathsf{canprogress}\,t\,s \qquad \left(\forall t'\,s'.\ \ t/s \longrightarrow t'/s' \ \Rightarrow\ \texttt{evalnpz}\,t'\,s'\,Q\right)}{\texttt{evalnpz}\,t\,s\,Q}$$

Specialization for divergence:

$$\frac{\mathsf{canprogress}\,t\,s \qquad \left(\forall t'\,s'.\ \ t/s \longrightarrow t'/s' \ \Rightarrow\ \texttt{divnz}\,t'\,s'\right)}{\texttt{divnz}\,t\,s}$$

Coinductive partial correctness in small-step

Partial correctness for non-deterministic semantics, coinductively:

$$\frac{Q\,v\,s}{\texttt{evalnpz}\,v\,s\,Q} \qquad \frac{\texttt{canprogress}\,t\,s \qquad \left(\forall t'\,s'.\ t/s \longrightarrow t'/s' \ \Rightarrow\ \texttt{evalnpz}\,t'\,s'\,Q\right)}{\texttt{evalnpz}\,t\,s\,Q}$$

Specialization for divergence:

$$\frac{\texttt{canprogress}\,t\,s \qquad \left(\forall t'\,s'.\ t/s \longrightarrow t'/s' \ \Rightarrow\ \texttt{divnz}\,t'\,s'\right)}{\texttt{divnz}\,t\,s}$$

Specialization for divergence, for deterministic semantics:

$$\frac{t/s \longrightarrow t'/s' \qquad \texttt{divdz}\,t'\,s'}{\texttt{divdz}\,t\,s}$$

5. Non-deterministic semantics in big-step

Part 1: total correctness

Big-step definitions for non-deterministic semantics

Conjecture: cannot handle non-determinism using $t/s \Downarrow v/s'$.

Big-step definitions for non-deterministic semantics

Conjecture: cannot handle non-determinism using $t/s \Downarrow v/s'$.

Idea: define the predicate "$\texttt{eval}\, s\, t\, Q$" inductively.

$t/s \Downarrow Q$: any evaluation of t/s terminates on a result that satisfies Q.

Multi-big-step semantics

$$\textsc{VAL}\ \frac{Q\,v\,s}{v/s \Downarrow Q} \qquad \textsc{APP}\ \frac{v_1 = \mu f.\lambda x.t_1 \qquad ([f \to v_1]\,[x \to v_2]\,t_1)/s \Downarrow Q}{(v_1\,v_2)/s \Downarrow Q}$$

$$\textsc{LET}\ \frac{t_1/s \Downarrow Q' \qquad \big(\forall v' s'.\ Q'\,v'\,s' \Rightarrow ([x \to v']\,t_2)/s' \Downarrow Q\big)}{(\mathsf{let}\,x = t_1\,\mathsf{in}\,t_2)/s \Downarrow Q}$$

Multi-big-step semantics

$$\text{VAL}\quad \frac{Q\,v\,s}{v/s \Downarrow Q} \qquad \text{APP}\quad \frac{v_1 = \mu f.\lambda x.t_1 \qquad ([f \to v_1]\,[x \to v_2]\,t_1)/s \Downarrow Q}{(v_1\,v_2)/s \Downarrow Q}$$

$$\text{LET}\quad \frac{t_1/s \Downarrow Q' \qquad \big(\forall v' s'.\ Q'\,v'\,s' \Rightarrow ([x \to v']\,t_2)/s' \Downarrow Q\big)}{(\mathsf{let}\,x = t_1\,\mathsf{in}\,t_2)/s \Downarrow Q}$$

$$\text{RAND}\quad \frac{n > 0 \qquad \big(\forall m.\ 0 \leqslant m < n \Rightarrow Q\,m\,s\big)}{(\mathsf{rand}\,n)/s \Downarrow Q} \qquad \text{GET}\quad \frac{p \in \mathsf{dom}\,s \qquad Q\,(s[p])\,s}{(\mathsf{get}\,p)/s \Downarrow Q}$$

$$\text{REF}\quad \frac{\forall p.\ p \notin \mathsf{dom}\,s \Rightarrow Q\,p\,(s[p := v])}{(\mathsf{ref}\,v)/s \Downarrow Q}$$

(`evalntb`)

Interpretation 1: generalization to a set of results

$t/s \Downarrow v/s'$ relates an input configuration t/s with one result (v, s').

$t/s \Downarrow Q$ relates an input configuration t/s with a set of results, described by Q.

Interpretation 1: generalization to a set of results

$t/s \Downarrow v/s'$ relates an input configuration t/s with one result (v, s').

$t/s \Downarrow Q$ relates an input configuration t/s with a set of results, described by Q.

Q has type "val $\to$ state $\to$ Prop", isomorphic to "(val $\times$ state) $\to$ Prop".

Interpretation 1: generalization to a set of results

$t/s \Downarrow v/s'$ relates an input configuration t/s with one result (v, s').

$t/s \Downarrow Q$ relates an input configuration t/s with a set of results, described by Q.

Q has type "val $\to$ state $\to$ Prop", isomorphic to "(val $\times$ state) $\to$ Prop".

Q is an over-approximation of the set of results produced by t/s.

Interpretation 2: inductively defined weakest-pre.

The predicate wp satisfies:

$$\{H\}\, t\, \{Q\} \quad \Longleftrightarrow \quad H \vdash \mathsf{wp}\, t\, Q \quad \Longleftrightarrow \quad \forall s.\, H\, s \Rightarrow \mathsf{wp}\, t\, Q\, s$$

Interpretation 2: inductively defined weakest-pre.

The predicate wp satisfies:

$$\{H\}\, t\, \{Q\} \quad \Longleftrightarrow \quad H \vdash \mathsf{wp}\, t\, Q \quad \Longleftrightarrow \quad \forall s.\, H\, s \Rightarrow \mathsf{wp}\, t\, Q\, s$$

Here:

$$\{H\}\, t\, \{Q\} \quad \equiv \quad \forall s.\, H\, s \Rightarrow (t/s \Downarrow Q)$$

Interpretation 2: inductively defined weakest-pre.

The predicate wp satisfies:

$$\{H\}\, t\, \{Q\} \quad \Longleftrightarrow \quad H \vdash \mathsf{wp}\, t\, Q \quad \Longleftrightarrow \quad \forall s.\, H\, s \Rightarrow \mathsf{wp}\, t\, Q\, s$$

Here:

$$\{H\}\, t\, \{Q\} \quad \equiv \quad \forall s.\, H\, s \Rightarrow (t/s \Downarrow Q)$$

For readability, let's write "$t/s \Downarrow Q$" as "$\texttt{evalntb}\, t\, s\, Q$". Difference:

$$\frac{\texttt{evalntb}\, t_1\, s\, Q' \quad \left(\forall v' s'.\ Q'\, v'\, s' \Rightarrow \texttt{evalntb}\, ([x \to v']\, t_2)\, s'\, Q\right)}{\texttt{evalntb}\, (\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2)\, s\, Q}\ \textsc{let}$$

$$\frac{\texttt{evalntb}\, t_1\, s\, (\lambda v' s'.\, \texttt{evalntb}\, ([x \to v']\, t_2)\, s'\, Q)}{\texttt{evalntb}\, (\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2)\, s\, Q}\ \textsc{let-wp-style}$$

Interpretation 2: inductively defined weakest-pre.

The predicate wp satisfies:

$$\{H\}\, t\, \{Q\} \quad \Longleftrightarrow \quad H \vdash \mathsf{wp}\, t\, Q \quad \Longleftrightarrow \quad \forall s.\, H\, s \Rightarrow \mathsf{wp}\, t\, Q\, s$$

Here:

$$\{H\}\, t\, \{Q\} \quad \equiv \quad \forall s.\, H\, s \Rightarrow (t/s \Downarrow Q)$$

For readability, let's write "$t/s \Downarrow Q$" as "$\texttt{evalntb}\, t\, s\, Q$". Difference:

$$\frac{\texttt{evalntb}\, t_1\, s\, Q' \quad \left(\forall v' s'.\ Q'\, v'\, s' \Rightarrow \texttt{evalntb}\, ([x \to v']\, t_2)\, s'\, Q\right)}{\texttt{evalntb}\, (\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2)\, s\, Q}\ \textsc{let}$$

$$\frac{\texttt{evalntb}\, t_1\, s\, (\lambda v' s'.\, \texttt{evalntb}\, ([x \to v']\, t_2)\, s'\, Q)}{\texttt{evalntb}\, (\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2)\, s\, Q}\ \textsc{let-wp-style}$$

The intermediate postcondition Q' in LET is required for Coq's approval.

Interpretation 3: generalized typing rules

Let's ignore states for this slide, and write $t \Downarrow Q$ in the form $\vdash t : Q$.

$$\textsc{let (substitution-based)}\quad \frac{\vdash t_1 : Q' \qquad \big(\forall v'.\ Q'\,v' \quad \Rightarrow \quad \vdash ([x \to v']\,t_2) : Q\big)}{\vdash (\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2) : Q}$$

Interpretation 3: generalized typing rules

Let's ignore states for this slide, and write $t \Downarrow Q$ in the form $\vdash t : Q$.

$$\textsc{let (substitution-based)}\quad \frac{\vdash t_1 : Q' \qquad \big(\forall v'.\ Q'\,v' \quad\Rightarrow\quad \vdash ([x \to v']\,t_2) : Q\big)}{\vdash (\mathsf{let}\,x = t_1\,\mathsf{in}\,t_2) : Q}$$

$$\textsc{let (environment-based)}\quad \frac{E \vdash t_1 : Q' \qquad \big(\forall v'.\ Q'\,v' \quad\Rightarrow\quad (E, x \mapsto v') \vdash t_2 : Q\big)}{E \vdash (\mathsf{let}\,x = t_1\,\mathsf{in}\,t_2) : Q}$$

Interpretation 3: generalized typing rules

Let's ignore states for this slide, and write $t \Downarrow Q$ in the form $\vdash t : Q$.

LET (SUBSTITUTION-BASED)

$$\frac{\vdash t_1 : Q' \qquad \big(\forall v'.\ Q'\,v' \quad \Rightarrow \quad \vdash ([x \to v']\,t_2) : Q\big)}{\vdash (\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2) : Q}$$

LET (ENVIRONMENT-BASED)

$$\frac{E \vdash t_1 : Q' \qquad \big(\forall v'.\ Q'\,v' \quad \Rightarrow \quad (E, x \mapsto v') \vdash t_2 : Q\big)}{E \vdash (\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2) : Q}$$

LET (AS TYPING RULE)

$$\frac{E \vdash t_1 : Q' \qquad (E, x : Q') \vdash t_2 : Q}{E \vdash (\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2) : Q}$$

Basic properties of the multi-big-step judgment

Consequence rule

$$\frac{t/s \Downarrow Q \qquad Q \vdash Q'}{t/s \Downarrow Q'}$$

Basic properties of the multi-big-step judgment

Consequence rule

$$\frac{t/s \Downarrow Q \qquad Q \vdash Q'}{t/s \Downarrow Q'}$$

Correctness w.r.t. the standard big-step judgment

$$\frac{t/s \Downarrow Q \qquad t/s \Downarrow v/s'}{Q\, v\, s'}$$

Maximal and minimal postconditions

Maximal postcondition: captures safety and termination
(all evaluations terminate, without getting stuck)

$$\text{terminates-safely}\, t\, s \quad \equiv \quad t/s \Downarrow (\lambda v s'.\, \text{True})$$

Maximal and minimal postconditions

Maximal postcondition: captures safety and termination
(all evaluations terminate, without getting stuck)

$$\mathsf{terminates\text{-}safely}\, t\, s \quad \equiv \quad t/s \Downarrow (\lambda v s'.\, \mathsf{True})$$

Minimal postcondition: characterizes the set of output

$$\mathsf{results}\, t\, s \quad \equiv \quad \lambda v s'.\, (t/s \Downarrow v/s') \quad : \quad \mathsf{val} \to \mathsf{state} \to \mathsf{Prop}$$

Maximal and minimal postconditions

Maximal postcondition: captures safety and termination
(all evaluations terminate, without getting stuck)

$$\mathsf{terminates\text{-}safely}\, t\, s \quad \equiv \quad t/s \Downarrow (\lambda v s'.\, \mathsf{True})$$

Minimal postcondition: characterizes the set of output

$$\mathsf{results}\, t\, s \quad \equiv \quad \lambda v s'.\, (t/s \Downarrow v/s') \quad : \quad \mathsf{val} \to \mathsf{state} \to \mathsf{Prop}$$

$$\frac{\mathsf{terminates\text{-}safely}\, t\, s}{t/s \Downarrow (\mathsf{results}\, t\, s)}\ \textsc{is-post} \qquad \frac{t/s \Downarrow Q}{\mathsf{results}\, t\, s \vdash_{\cdot} Q}\ \textsc{is-minimal}$$

Interpretation in the case of deterministic semantics

For a deterministic semantics, how does $t/s \Downarrow v/s'$ relate to $t/s \Downarrow Q$?

Interpretation in the case of deterministic semantics

For a deterministic semantics, how does $t/s \Downarrow v/s'$ relate to $t/s \Downarrow Q$?

Equivalence:

$$\text{deterministic} \quad \Rightarrow \quad \big(t/s \Downarrow v/s'\big) \iff \big(t/s \Downarrow \text{exactly}\, v\, s'\big)$$

where

$$\text{exactly}\, v\, s' \quad \equiv \quad \lambda xy.\ (x = v \wedge y = s')$$

Deriving Hoare Logic reasoning rules

Hoare reasoning rule:

$$\frac{\{H\}\ t_1\ \{Q'\} \qquad \left(\forall v'.\ \{Q'\ v'\}\ ([x \to v']\, t_2)\ \{Q\}\right)}{\{H\}\ (\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2)\ \{Q\}}\ \textsc{hoare-let}$$

Deriving Hoare Logic reasoning rules

Hoare reasoning rule:

$$\frac{\{H\}\ t_1\ \{Q'\} \qquad \big(\forall v'.\ \ \{Q'\ v'\}\ ([x \to v']\, t_2)\ \{Q\}\big)}{\{H\}\ (\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2)\ \{Q\}}\ \textsc{hoare-let}$$

Proof:

$$\{H\}\ t\ \{Q\} \quad \equiv \quad \forall s.\ \ H\ s \ \Rightarrow\ \ t/s \Downarrow Q$$

$$\frac{t_1/s \Downarrow Q' \qquad \big(\forall v' s'.\ \ Q'\ v'\ s' \Rightarrow ([x \to v']\, t_2)/s' \Downarrow Q\big)}{(\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2)/s \Downarrow Q}\ \textsc{let}$$

```
Proof. intros * M1 M2 h K. applys evalntb_let; eauto. Qed.
```

5. Non-deterministic semantics in big-step

Part 2: partial correctness

Partial correctness in multi-big-step

Coinductive intepretation of the exact same set of rules:
$t/s \Downarrow^{\mathsf{co}} Q$ asserts that every possible evaluation either diverges or terminates on a result satisfying Q, but does not get stuck.

$$\frac{Q\,v\,s}{v/s \Downarrow^{\mathsf{co}} Q} \qquad \frac{v_1 = \mu f.\lambda x.t_1 \qquad ([f \to v_1]\,[x \to v_2]\,t_1)/s \Downarrow^{\mathsf{co}} Q}{(v_1\,v_2)/s \Downarrow^{\mathsf{co}} Q}$$

$$\frac{t_1/s \Downarrow^{\mathsf{co}} Q' \qquad \big(\forall v' s'.\ Q'\,v'\,s' \Rightarrow ([x \to v']\,t_2)/s' \Downarrow^{\mathsf{co}} Q\big)}{(\mathsf{let}\,x = t_1\,\mathsf{in}\,t_2)/s \Downarrow^{\mathsf{co}} Q}$$

$$\frac{n > 0 \qquad \big(\forall m.\ 0 \leqslant m < n \Rightarrow Q\,m\,s\big)}{(\mathsf{rand}\,n)/s \Downarrow^{\mathsf{co}} Q} \qquad \frac{p \in \mathsf{dom}\,s \qquad Q\,(s[p])\,s}{(\mathsf{get}\,p)/s \Downarrow^{\mathsf{co}} Q}$$

$$\frac{\forall p.\ p \notin \mathsf{dom}\,s \Rightarrow Q\,p\,(s[p := v])}{(\mathsf{ref}\,v)/s \Downarrow^{\mathsf{co}} Q}$$

Zoom on the divergence of let-expressions

$t/s \Uparrow^{\mathsf{co}}$ asserts the divergence of one possible execution.

$$\textsc{div-let-1}\ \frac{t_1/s \Uparrow^{\mathsf{co}}}{(\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2)/s \Uparrow^{\mathsf{co}}} \qquad \textsc{div-let-2}\ \frac{t_1/s \Downarrow v_1/s' \qquad ([x \to v_1]\, t_2)/s' \Uparrow^{\mathsf{co}}}{(\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2)/s \Uparrow^{\mathsf{co}}}$$

Zoom on the divergence of let-expressions

$t/s \Uparrow^{\mathsf{co}}$ asserts the divergence of one possible execution.

$$\textsc{div-let-1}\ \frac{t_1/s \Uparrow^{\mathsf{co}}}{(\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2)/s \Uparrow^{\mathsf{co}}} \qquad \textsc{div-let-2}\ \frac{t_1/s \Downarrow v_1/s' \qquad ([x \to v_1]\, t_2)/s' \Uparrow^{\mathsf{co}}}{(\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2)/s \Uparrow^{\mathsf{co}}}$$

$t/s \Downarrow^{\mathsf{co}} (\lambda v s'.\, \mathsf{False})$ asserts the divergence of every possible execution.

$$\textsc{let}\ \frac{t_1/s \Downarrow^{\mathsf{co}} Q' \qquad \left(\forall v' s'.\ Q'\, v'\, s' \Rightarrow ([x \to v']\, t_2)/s' \Downarrow^{\mathsf{co}} Q\right)}{(\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2)/s \Downarrow^{\mathsf{co}} Q}$$

Summary of multi-big-step definitions

Big-step total correctness: (`evalntb`)
essentially an inductive definition of wp.

$$^{total}\{H\}\,t\,\{Q\} \quad \equiv \quad \forall s.\ H\,s \ \Rightarrow\ t/s \Downarrow Q$$

Big-step partial correctness: (`evalnpb`)
co-inductive interpretation of the same rules.

$$^{partial}\{H\}\,t\,\{Q\} \quad \equiv \quad \forall s.\ H\,s \ \Rightarrow\ t/s \Downarrow^{\mathsf{co}} Q$$

Big-step divergence: (`divnb`)
partial correctness with empty postcondition.

$$\mathsf{diverges}\,t\,s \quad \equiv \quad t/s \Downarrow^{\mathsf{co}} (\lambda v s'.\,\mathsf{False})$$

Summary of multi-big-step definitions

Big-step total correctness: (`evalntb`)
essentially an inductive definition of wp.

$$^{total}\{H\}\,t\,\{Q\} \quad \equiv \quad \forall s.\ H\,s \ \Rightarrow\ t/s \Downarrow Q$$

Big-step partial correctness: (`evalnpb`)
co-inductive interpretation of the same rules.

$$^{partial}\{H\}\,t\,\{Q\} \quad \equiv \quad \forall s.\ H\,s \ \Rightarrow\ t/s \Downarrow^{\mathsf{co}} Q$$

Big-step divergence: (`divnb`)
partial correctness with empty postcondition.

$$\mathsf{diverges}\,t\,s \quad \equiv \quad t/s \Downarrow^{\mathsf{co}} (\lambda v s'.\,\mathsf{False})$$

$$\mathsf{diverges}\,t\,s \quad \Longleftrightarrow \quad \forall Q.\ (t/s \Downarrow^{\mathsf{co}} Q)$$

Summary table

		det.(**d**)	non-det.(**n**)	complete(**c**)
total(**t**)	big (**b**)	`evaldtb`	`evalntb`	`(evalntb)`
	small-ind.(**z**)	`evaldtz`	`evalntz`	`evalctz`
	small(**s**)	`evaldts`	–	–
partial(**p**)	big (**b**)	`evaldpb`	`evalnpb`	`evalcpb`
	small-co.(**z**)	`evaldpz`	`evalnpz`	`evalcpz`
	small(**s**)	`(evalnps)`	`evalnps`	`evalcps`
diverge(**d**)	big (**b**)	`divdb`	`divnb`	`(divnb)`
	small-co.(**z**)	`divdz`	`divnz`	`divcz`
	small(**s**)	`(divns)`	`divns`	`(divns)`

In parenthesis: no simpler definition than the non-deterministic one.

Equivalences proved

Summary table

		det.(**d**)	non-det.(**n**)	complete(**c**)
total(**t**)	big (**b**)	`evaldtb`	`evalntb`	`(evalntb)`
	small-ind.(**z**)	`evaldtz`	`evalntz`	`evalctz`
	small(**s**)	`evaldts`	–	–
partial(**p**)	big (**b**)	`evaldpb`	`evalnpb`	`evalcpb`
	small-co.(**z**)	`evaldpz`	`evalnpz`	`evalcpz`
	small(**s**)	`(evalnps)`	`evalnps`	`evalcps`
diverge(**d**)	big (**b**)	`divdb`	`divnb`	`(divnb)`
	small-co.(**z**)	`divdz`	`divnz`	`divcz`
	small(**s**)	`(divns)`	`divns`	`(divns)`

In parenthesis: no simpler definition than the non-deterministic one.

Language considered

$$
\begin{array}{lcl}
v & := & n \mid p \mid b \mid \mu f.\lambda x.t \mid (+) \mid \mathsf{ref} \mid \mathsf{get} \mid \mathsf{set} \mid \mathsf{free} \mid \mathsf{rand} \mid \mathsf{err} \\
t & := & x \mid v \mid \mu f.\lambda x.t \mid t_1\, t_2 \mid \mathsf{let}\, x = t_1\, \mathsf{in}\, t_2 \mid \mathsf{if}\, t_0\, \mathsf{then}\, t_1\, \mathsf{else}\, t_2
\end{array}
$$

Features:

- Semantics defined for programs in A-normal form: only one context.
- Flag deterministic: disables rand, and makes ref pick smallest fresh.
- Flag complete: adds rules for error propagation (err) on stuck terms.

Language considered

$$
\begin{array}{lcl}
v & := & n \mid p \mid b \mid \mu f.\lambda x.t \mid (+) \mid \mathsf{ref} \mid \mathsf{get} \mid \mathsf{set} \mid \mathsf{free} \mid \mathsf{rand} \mid \mathsf{err} \\
t & := & x \mid v \mid \mu f.\lambda x.t \mid t_1\, t_2 \mid \mathsf{let}\, x = t_1\, \mathsf{in}\, t_2 \mid \mathsf{if}\, t_0\, \mathsf{then}\, t_1\, \mathsf{else}\, t_2
\end{array}
$$

Features:

- Semantics defined for programs in A-normal form: only one context.
- Flag deterministic: disables rand, and makes ref pick smallest fresh.
- Flag complete: adds rules for error propagation (err) on stuck terms.

```
| step_let_err : ∀s1 x t2,
    complete →
    step s1 (trm_let x err t2) s1 err.
| step_let : ∀s x t2 v1,
    (complete → v1 ≠err) →
    step s (trm_let x v1 t2) s (subst x v1 t2)
| step_stuck : ∀s1 t1,
    complete →
    stuck s1 t1 →
    step s1 t1 s1 err
```

Conclusion

- This work covers the spectrum of definitions for Hoare triples.
- The multi-big-step judgment $t/s \Downarrow Q$ is:
  1. the natural generalization of $t/s \Downarrow v/s'$ to multiple results
  2. another way to look at the weakest-precondition predicate
  3. a strong form of typing judgment

Conclusion

- This work covers the spectrum of definitions for Hoare triples.
- The multi-big-step judgment $t/s \Downarrow Q$ is:
  1. the natural generalization of $t/s \Downarrow v/s'$ to multiple results
  2. another way to look at the weakest-precondition predicate
  3. a strong form of typing judgment
- $t/s \Downarrow Q$ and $t/s \Downarrow^{\text{co}} Q$ define the semantics of terminating and diverging programs. Hoare Logic rules can be trivially derived.

Conclusion

- This work covers the spectrum of definitions for Hoare triples.
- The multi-big-step judgment $t/s \Downarrow Q$ is:
  1. the natural generalization of $t/s \Downarrow v/s'$ to multiple results
  2. another way to look at the weakest-precondition predicate
  3. a strong form of typing judgment
- $t/s \Downarrow Q$ and $t/s \Downarrow^{\mathsf{co}} Q$ define the semantics of terminating and diverging programs. Hoare Logic rules can be trivially derived.

Thanks!

Multi-big-step with more than two premises

Reformulation of the let-rule:

$$\frac{t_1/s \Downarrow Q' \qquad \big(\forall v' s'.\ Q'\, v'\, s' \Rightarrow ([x \to v']\, t_2)/s' \Downarrow Q\big)}{(\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2)/s \Downarrow Q}\ \textsc{let}$$

$$\frac{\big(\exists Q'.\quad t_1/s \Downarrow Q' \ \wedge\ \big(\forall v' s'.\ Q'\, v'\, s' \Rightarrow ([x \to v']\, t_2)/s' \Downarrow Q\big)\big)}{(\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2)/s \Downarrow Q}\ \textsc{let}',$$

Multi-big-step with more than two premises

Reformulation of the let-rule:

$$\frac{t_1/s \Downarrow Q' \qquad \big(\forall v' s'.\ Q'\, v'\, s' \Rightarrow ([x \to v']\, t_2)/s' \Downarrow Q\big)}{(\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2)/s \Downarrow Q}\ \textsc{let}$$

$$\frac{\big(\exists Q'.\quad t_1/s \Downarrow Q' \ \wedge\ \big(\forall v' s'.\ Q'\, v'\, s' \Rightarrow ([x \to v']\, t_2)/s' \Downarrow Q\big)\big)}{(\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2)/s \Downarrow Q}\ \textsc{let}',$$

Rule for applications not in A-normal form:

$$\frac{\begin{array}{c}\exists Q_1.\quad t_1/s_0 \Downarrow Q_1 \ \wedge\ \big(\forall v_1 s_1.\ Q_1\, v_1\, s_1 \Rightarrow \\ \exists Q_2.\quad t_2/s' \Downarrow Q_2 \ \wedge\ \big(\forall v_2 s_2.\ Q_2\, v_2\, s_2 \Rightarrow \\ \forall f x t_3 s'.\ v_1 = \mu f.\lambda x.t_3 \Rightarrow ([f \to v_1]\, [x \to v_2]\, t_3)/s_2 \Downarrow Q\big)\big)\end{array}}{(t_1\, t_2)/s_0 \Downarrow Q}\ \textsc{app}',$$

Multi-big-step with more than two premises

Reformulation of the let-rule:

$$\frac{t_1/s \Downarrow Q' \qquad (\forall v' s'.\ Q'\, v'\, s' \Rightarrow ([x \to v']\, t_2)/s' \Downarrow Q)}{(\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2)/s \Downarrow Q}\ \textsc{let}$$

$$\frac{(\exists Q'.\quad t_1/s \Downarrow Q' \ \wedge\ (\forall v' s'.\ Q'\, v'\, s' \Rightarrow ([x \to v']\, t_2)/s' \Downarrow Q))}{(\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2)/s \Downarrow Q}\ \textsc{let'},$$

Rule for applications not in A-normal form:

$$\frac{\begin{array}{c}\exists Q_1.\quad t_1/s_0 \Downarrow Q_1 \ \wedge\ (\forall v_1 s_1.\ Q_1\, v_1\, s_1 \Rightarrow \\ \exists Q_2.\quad t_2/s' \Downarrow Q_2 \ \wedge\ (\forall v_2 s_2.\ Q_2\, v_2\, s_2 \Rightarrow \\ \forall f x t_3 s'.\ v_1 = \mu f.\lambda x.t_3 \Rightarrow ([f \to v_1]\, [x \to v_2]\, t_3)/s_2 \Downarrow Q))\end{array}}{(t_1\, t_2)/s_0 \Downarrow Q}\ \textsc{app'},$$

Disclaimer: Coq does not generate the right induction principle for free.
Using pretty-big-step semantics ensures at most two premises.

Equivalence of multi-big-step, total correctness

Recall:

$$\frac{Q\,v\,s}{\texttt{evalntz}\,v\,s\,Q} \qquad \frac{\mathsf{canprogress}\,t\,s \quad \left(\forall t' s'.\ \ t/s \longrightarrow t'/s' \ \Rightarrow\ \texttt{evalntz}\,t'\,s'\,Q\right)}{\texttt{evalntz}\,t\,s\,Q}$$

Equivalence of multi-big-step, total correctness

Recall:

$$\frac{Q\,v\,s}{\texttt{evalntz}\,v\,s\,Q} \qquad \frac{\texttt{canprogress}\,t\,s \quad \left(\forall t' s'.\ t/s \longrightarrow t'/s' \ \Rightarrow\ \texttt{evalntz}\,t'\,s'\,Q\right)}{\texttt{evalntz}\,t\,s\,Q}$$

From `evalntb` to `evalntz`, we need big-step results for `evalntz`, e.g.:

$$\frac{\texttt{evalntz}\,t_1\,s\,Q' \qquad \left(\forall v' s'.\ Q'\,v'\,s' \ \Rightarrow\ \texttt{evalntz}\,([x \to v']\,t_2)\,s'\,Q\right)}{\texttt{evalntz}\,(\mathsf{let}\,x = t_1\,\mathsf{in}\,t_2)\,s\,Q}$$

Equivalence of multi-big-step, total correctness

Recall:

$$\frac{Q\,v\,s}{\texttt{evalntz}\,v\,s\,Q} \qquad \frac{\textsf{canprogress}\,t\,s \quad \big(\forall t' s'.\ t/s \longrightarrow t'/s' \ \Rightarrow\ \texttt{evalntz}\,t'\,s'\,Q\big)}{\texttt{evalntz}\,t\,s\,Q}$$

From `evalntb` to `evalntz`, we need big-step results for `evalntz`, e.g.:

$$\frac{\texttt{evalntz}\,t_1\,s\,Q' \qquad \big(\forall v' s'.\ Q'\,v'\,s' \ \Rightarrow\ \texttt{evalntz}\,([x \to v']\,t_2)\,s'\,Q\big)}{\texttt{evalntz}\,(\textsf{let}\,x = t_1\,\textsf{in}\,t_2)\,s\,Q}$$

From `evalntz` to `evalntb`, we need a composition lemma.

$$\frac{\textsf{canprogress}\,t\,s \qquad \big(\forall t' s'.\ (t/s \longrightarrow t'/s') \ \Rightarrow\ (t'/s' \Downarrow Q)\big)}{t/s \Downarrow Q}$$

Equivalence of multi-big-step, partial correctness

From `evalnpb` to `evalnpz`, we need an inversion lemma to pull one step.

$$t/s \Downarrow^{\mathsf{co}} Q \quad \Rightarrow \quad \begin{array}{ll} & (\exists v.\ t = \mathsf{val}\, v \ \wedge\ Q\, v\, s\ ) \\ \vee & \left(\begin{array}{ll} & (\mathsf{canprogress}\, t\, s) \\ \wedge & (\forall t' s'.\ (t/s \longrightarrow t'/s') \ \Rightarrow\ (t'/s' \Downarrow^{\mathsf{co}} Q)) \end{array}\right) \end{array}$$

Equivalence of multi-big-step, partial correctness

From `evalnpb` to `evalnpz`, we need an inversion lemma to pull one step.

$$t/s \Downarrow^{\mathsf{co}} Q \quad \Rightarrow \quad \begin{array}{ll} & (\exists v.\ t = \mathsf{val}\, v \ \wedge\ Q\, v\, s\ ) \\ \vee & \left(\begin{array}{ll} & (\mathsf{canprogress}\, t\, s) \\ \wedge & (\forall t' s'.\ (t/s \longrightarrow t'/s') \Rightarrow (t'/s' \Downarrow^{\mathsf{co}} Q)) \end{array}\right) \end{array}$$

From `evalnpz` to `evalnpb`, we need an inversion lemma for let-bindings.

$$\begin{array}{l} \mathtt{evalntz}\,(\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2)\, s\, Q \quad \Rightarrow \\ \quad \exists Q'.\quad \mathtt{evalntz}\, t_1\, s\, Q' \\ \qquad\quad \wedge\quad (\forall v' s'.\ Q\, v'\, s' \Rightarrow \mathtt{evalntz}\,([x \to v']\, t_2)\, s'\, Q) \end{array}$$

(The witness for Q' is the minimal postcondition, namely "results $t_1\, s$".)

Refinement of the lemmas for a complete semantics

Introduction lemma for let-bindings for `evalntz`

$$\frac{\begin{array}{c}\texttt{evalntz}\, t_1\, s\, Q' \\ \left(\forall v' s'.\ Q'\, v'\, s' \ \wedge\ v' \neq \mathsf{err} \ \Rightarrow\ \texttt{evalntz}\, ([x \to v']\, t_2)\, s'\, Q\right) \\ \left(\forall s'.\ Q'\, \mathsf{err}\, s \ \Rightarrow\ Q\, \mathsf{err}\, s\right)\end{array}}{\texttt{evalntz}\, (\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2)\, s\, Q}$$

Inversion lemma for let-bindings for `evalnpz`

$$\texttt{evalntz}\, (\mathsf{let}\, x = t_1\, \mathsf{in}\, t_2)\, s\, Q \quad \Rightarrow$$

$$\begin{array}{lll} \exists Q'. & \texttt{evalntz}\, t_1\, s\, Q' \\ \wedge & \left(\forall v' s'.\ Q\, v'\, s' \ \wedge\ v' \neq \mathsf{err} \ \Rightarrow\ \texttt{evalntz}\, ([x \to v']\, t_2)\, s'\, Q\right) \\ \wedge & \left(\forall s'.\ Q'\, \mathsf{err}\, s \ \Rightarrow\ Q\, \mathsf{err}\, s\right) \end{array}$$