A Hoare logic for the coinductive trace-based big-step semantics of While

Keiko Nakata
Institute of Cybernetics, Tallinn University of Technology
Joint work with T. Uustalu

October 2009

Motivation

There are important programs that are not supposed to terminate, e.g. operating systems and data base systems.

Our motivation is to set up a foundational framework in a constructive type theory that accounts for both terminating and diverging program runs.

Applications include

- certified compilers, program transformations
- information flow analysis

What we have done

We study the While language.

We have devised:

- trace-based big-step relational semantics, as well as small-step relational semantics and big-step & small-step functional semantics
  They are all defined coinductively and equivalent constructively.
- Hoare logic, sound and complete with respect to the semantics

All results are formalized fully constructively in Coq.

The While language

$$\begin{array}{rcl} x, y, z & \in & \textit{Variables} \\ e & \in & \textit{Expressions} \\ v & \in & \textit{Integers} \\ \sigma & \in & \textit{Variables} \rightarrow \textit{Integers} \end{array}$$

$$\begin{array}{llcl} \textit{statement} & s & ::= & \text{skip} \mid s_0; s_1 \mid x := e \\ & & \mid & \text{if } e \text{ then } s_t \text{ else } s_f \mid \text{while } e \text{ do } s_t \end{array}$$

Notations

The While language

$\sigma[x \mapsto v]$ denotes the update of σ with v at x.

$[\![e]\!]\sigma$ evaluates e in the state σ.
E.g. $[\![x + y]\!]\{x \mapsto 2, x \mapsto 2\} = 4$

$\sigma \models e$ denotes that e evaluates to truth (non-zero) in σ.
E.g $\{x \mapsto 2, x \mapsto 2\} \models x + y$

$\sigma \not\models e$ denotes that e evaluates to falsity (zero) in σ.
E.g $\{x \mapsto 2, x \mapsto 2\} \not\models x - y$

Traces

Traces $\tau \in \mathit{trace}$ are possibly infinite non-empty sequences of states, defined coinductively by:

$$\frac{}{\langle \sigma \rangle \in \mathit{trace}} \qquad \frac{\tau \in \mathit{trace}}{\sigma :: \tau \in \mathit{trace}}$$

We define bisimilarity (equivalence relation) between traces, $\tau \approx \tau'$, coinductively by:

$$\frac{}{\langle \sigma \rangle \approx \langle \sigma \rangle} \qquad \frac{\tau \approx \tau'}{\sigma :: \tau \approx \sigma :: \tau'}$$

We think of bisimilar traces as equal, i.e. traces as a setoid with bisimilarity as the equivalence relation.

Finiteness and infiniteness

We define inductively a trace predicate *finite* τ stating that τ is finite:

$$\frac{}{\textit{finite}\ \langle\sigma\rangle} \qquad \frac{\textit{finite}\ \tau}{\textit{finite}\ \sigma :: \tau}$$

We define coinductively a trace predicate *infinite* τ stating that τ is infinite:

$$\frac{\textit{infinite}\ \tau}{\textit{infinite}\ \sigma :: \tau}$$

We can define infiniteness constructively, not as negation of finiteness.

Finiteness and infiniteness (2)

Working in a constructive logic, our trace predicates have a rich structure.

- $\neg$ *finite* $\models$ *infinite*
- $\neg$ *infinite* $\models$ *finite* is not probable constructively.
  (But is provably classically.)

ref.
Constructive logic does not have the law of excluded middle

$$\forall P : Prop, P \vee \neg P$$

The evaluation $(s, \sigma) \Rightarrow \tau$ expresses that running a statement s from a state σ produces a trace τ.

E.g.

$(\mathrm{skip}, \sigma) \Rightarrow \langle \sigma \rangle$

$(x := 1 + 3; y := 2, (0, 0)) \Rightarrow (0, 0) :: (4, 0) :: \langle (4, 2) \rangle$

$(\mathrm{if}\ x = 0\ \mathrm{then}\ y := 1\ \mathrm{else}\ y := 2, (1, 0)) \Rightarrow$
$(1, 0) :: (1, 0) :: \langle (1, 2) \rangle$

$(\mathrm{while\ true\ do\ skip}, \sigma) \Rightarrow \sigma :: \sigma :: \sigma :: \ldots$

The judgment forms

Big-step semantics

$(s, \sigma) \Rightarrow \tau$ is defined by mutual coinduction together with the extended evaluation $(s, \tau) \stackrel{*}{\Rightarrow} \tau'$.

$(s, \tau) \stackrel{*}{\Rightarrow} \tau'$ expresses that running a statement s from the last state (if it exists) of an already accumulated trace τ results in a total trace τ'. Or:

$$\frac{(s, \sigma) \Rightarrow \tau}{(s, \langle \sigma \rangle) \stackrel{*}{\Rightarrow} \tau} \qquad \frac{(s, \tau) \stackrel{*}{\Rightarrow} \tau'}{(s, \sigma :: \tau) \stackrel{*}{\Rightarrow} \sigma :: \tau'}$$

E.g.

$(x := 1 + 3; y := 2, (0, 0) :: \langle (0, 1) \rangle) \stackrel{*}{\Rightarrow}$
$(0, 0) :: (0, 1) :: (4, 1) :: \langle (4, 2) \rangle$

Inference rules

Big-step semantics

$$\frac{}{(x := e, \sigma) \Rightarrow \sigma :: \langle \sigma[x \mapsto [\![e]\!]\sigma] \rangle}$$

$$\frac{}{(\mathrm{skip}, \sigma) \Rightarrow \langle \sigma \rangle} \qquad \frac{(s_0, \sigma) \Rightarrow \tau \quad (s_1, \tau) \overset{*}{\Rightarrow} \tau'}{(s_0; s_1, \sigma) \Rightarrow \tau'}$$

$$\frac{\sigma \models e \quad (s_t, \sigma :: \langle \sigma \rangle) \overset{*}{\Rightarrow} \tau}{(\text{if } e \text{ then } s_t \text{ else } s_f, \sigma) \Rightarrow \tau} \qquad \frac{\sigma \not\models e \quad (s_f, \sigma :: \langle \sigma \rangle) \overset{*}{\Rightarrow} \tau}{(\text{if } e \text{ then } s_t \text{ else } s_f, \sigma) \Rightarrow \tau}$$

$$\frac{\sigma \models e \quad (s_t, \sigma :: \langle \sigma \rangle) \overset{*}{\Rightarrow} \tau \quad (\text{while } e \text{ do } s_t, \tau) \overset{*}{\Rightarrow} \tau'}{(\text{while } e \text{ do } s_t, \sigma) \Rightarrow \tau'}$$

$$\frac{\sigma \not\models e}{(\text{while } e \text{ do } s_t, \sigma) \Rightarrow \sigma :: \langle \sigma \rangle}$$

$$\frac{(s, \sigma) \Rightarrow \tau}{(s, \langle \sigma \rangle) \overset{*}{\Rightarrow} \tau} \qquad \frac{(s, \tau) \overset{*}{\Rightarrow} \tau'}{(s, \sigma :: \tau) \overset{*}{\Rightarrow} \sigma :: \tau'}$$

Hoare logic

Our Hoare-triple $\{U\}\ s\ \{P\}$ consists of

U : predicate on states

s : statement

P : predicate on traces

$\{U\}\ s\ \{P\}$ means that running a statement s from a initial state σ satisfying U produces a trace τ satisfying P.

$\{x = 3\}$ while $x = 0$ do $x := x - 1$ $\{\mathit{finite}\}$

$\{x = -3\}$ while $x = 0$ do $x := x - 1$ $\{\mathit{infinite}\}$

Notations

U, V : state predicates
P, Q : trace predicates

$\sigma \models U$ expresses that σ satisfies U.
$\tau \models P$ expresses that τ satisfies P.

Logical consequences and equivalence:

$$\frac{\forall\sigma\,(\sigma \models U \rightarrow \sigma \models V)}{U \models V} \quad \frac{\forall\tau\,(\tau \models P \rightarrow \tau \models Q)}{P \models Q} \quad \frac{P \models Q \quad Q \models P}{P \Leftrightarrow Q}$$

Assertions

$$\begin{array}{c} \sigma \models U \\ \hline \langle\sigma\rangle \models \langle U \rangle \end{array} \quad \begin{array}{c} \sigma \models U \\ \hline \sigma :: \langle\sigma\rangle \models \langle U \rangle^2 \end{array} \quad \begin{array}{c} \sigma \models U \\ \hline \sigma :: (\sigma[x \mapsto e]) \models U[x \mapsto e] \end{array}$$

$$\begin{array}{c} \langle\sigma\rangle \models P \\ \hline\hline \langle\sigma\rangle \models_{\langle\sigma\rangle} P \end{array} \quad \begin{array}{c} \sigma :: \tau \models P \\ \hline\hline \sigma :: \tau \models_{\langle\sigma\rangle} P \end{array} \quad \begin{array}{c} \tau' \models_{\tau} P \\ \hline\hline \sigma :: \tau' \models_{\sigma::\tau} P \end{array}$$

$$\begin{array}{c} \tau' \models P \quad \tau \models_{\tau'} Q \\ \hline \tau \models P ** Q \end{array} \quad \begin{array}{c} \tau \models \langle \text{true} \rangle \\ \hline\hline \tau \models P^{\dagger} \end{array} \quad \begin{array}{c} \tau' \models P \quad \tau \models_{\tau'} P^{\dagger} \\ \hline\hline \tau \models P^{\dagger} \end{array}$$

$$\begin{array}{c} \tau \models P \quad \tau \downarrow \sigma \\ \hline \sigma \models \mathit{Last}\ P \end{array}$$

Singleton operator $\langle U \rangle$

$\langle U \rangle$ is a trace predicate that is true of a singleton trace given by a state satisfying U:

$$\frac{\sigma \models U}{\langle \sigma \rangle \models \langle U \rangle}$$

$\langle \mathrm{true} \rangle$ is true of any singleton trace.

$\langle U \rangle^2$ is a trace predicate that is true of a doubleton trace of an identical state satisfying U:

$$\frac{\sigma \models U}{\sigma :: \langle \sigma \rangle \models \langle U \rangle^2}$$

Update operator $U[x \mapsto e]$

Assertions

$U[x \mapsto e]$ is a trace predicate that is the strong postcondition of $x := e$ for the precondition U:

$$\frac{\sigma \models U}{\sigma :: \langle \sigma[x \mapsto e] \rangle \models U[x \mapsto e]}$$

Chop operator $P ** Q$

Assertions

Roughly, $\tau \models P ** Q$ holds when τ is split into two parts τ' and τ'' such that the last state of τ' is the first state of τ'' and the prefix τ' (resp. the postfix τ'') satisfies P (resp. Q):

$$\frac{\tau' \models P \quad \tau \models_{\tau'} Q}{\tau \models P ** Q}$$

$$\frac{\langle\sigma\rangle \models P}{\langle\sigma\rangle \models_{\langle\sigma\rangle} P} \qquad \frac{\sigma :: \tau \models P}{\sigma :: \tau \models_{\langle\sigma\rangle} P} \qquad \frac{\tau' \models_{\tau} P}{\sigma :: \tau' \models_{\sigma::\tau} P}$$

$\tau \models_{\tau'} P$ first traverses τ', which must be a prefix of τ, then checks validity of P against the postfix.

In particular, $\tau \models_{\tau'} P$ necessarily holds when τ' is infinite.

Chop operator $P ** Q$ (2)

The definition of $\tau \models P ** Q$ has the desirable property that if *infinite* τ and $\tau \models P$ then $\tau \models P ** Q$ for any Q.

In particular, we have:

- $P **$ false $\Leftrightarrow P \wedge$ *infinite*.
  As a special case: true $**$ false $\Leftrightarrow$ *infinite*.
- $\langle U \rangle ** P \models P$

Iteration operator $P^\dagger$

Assertions

$P^\dagger$ is a trace predicate that is true of a trace that is zero or possibly infinite concatenations of traces, each of which satisfies P:

$$\frac{\tau \models \langle \text{true} \rangle}{\tau \models P^\dagger} \qquad \frac{\tau' \models P \quad \tau \models_{\tau'} P^\dagger}{\tau \models P^\dagger}$$

We have:

$$P^\dagger \Leftrightarrow \langle \text{true} \rangle \vee (P ** P^\dagger)$$

Last P is a state predicate that is true of a state that can be the last state of a finite trace satisfying *P*:

$$\frac{\tau \models P \quad \tau \downarrow \sigma}{\sigma \models \mathit{Last}\ P}$$

We have:

- $\mathit{Last\ infinite} \Leftrightarrow$ false
- $P \Leftrightarrow P ** \langle \mathit{Last}\ P \rangle$

Inference rules of the Hoare logic

$$\frac{}{\{U\}\ x := e\ \{U[x \mapsto e]\}} \quad \frac{}{\{U\}\ \text{skip}\ \{\langle U \rangle\}}$$

$$\frac{\{U\}\ s_0\ \{P\} \quad \{\mathit{Last}\ P\}\ s_1\ \{Q\}}{\{U\}\ s_0; s_1\ \{P \ast\ast\ Q\}}$$

$$\frac{\{e \wedge U\}\ s_t\ \{P\} \quad \{\neg e \wedge U\}\ s_f\ \{P\}}{\{U\}\ \text{if}\ e\ \text{then}\ s_t\ \text{else}\ s_f\ \{\langle U \rangle^2 \ast\ast\ P\}}$$

$$\frac{U \models I \quad \{e \wedge I\}\ s_t\ \{P \ast\ast\ \langle I \rangle\}}{\{U\}\ \text{while}\ e\ \text{do}\ s_t\ \{\langle U \rangle^2 \ast\ast\ (\langle e \rangle \ast\ast\ P \ast\ast\ \langle I \rangle^2)^\dagger \ast\ast\ \langle I \wedge \neg e \rangle\}}$$

$$\frac{U \models U' \quad \{U'\}\ s\ \{P'\} \quad P' \models P}{\{U\}\ s\ \{P\}}$$

Soundness and Completeness

Proposition (Soundness)

For any s, U, P, σ, τ, *if* $\{U\}\ s\ \{P\}$ *and* $\sigma \models U$ *and* $(s, \sigma) \Rightarrow \tau$, *then* $\tau \models P$.

Proposition (Completeness)

For any s, U, P, *if for all* σ, τ, $\sigma \models U$ *and* $(s, \sigma) \Rightarrow \tau$ *imply* $\tau \models P$, *then* $\{U\}\ s\ \{P\}$.

Embedding of the standard Hoare logics

Proposition (Partial correctness)

For any u, s *and* v *if* $\{u\}\ s\ \{v\}$ *is derivable in the partial correctness Hoare logic, then* $\{u\}\ s\ \{\text{true} ** \langle v\rangle\}$.

Proof.

By induction on the derivation of $\{u\}\ s\ \{v\}$. □

Proposition (Total correctness)

For any u, s *and* v *if* $\{u\}\ s\ \{v\}$ *is derivable in the total correctness Hoare logic, then* $\{u\}\ s\ \{(\text{true} ** \langle v\rangle) \wedge \mathit{finite}\}$.

Proof.

By induction on the derivation of $\{u\}\ s\ \{v\}$. □

Unbounded total search

Example

Variable $B : nat \to bool$
Axiom $B\_noncontradictory$: $\neg(\forall n, \neg B\ n)$
Let s be

$$\text{while } \neg(B\ x) \text{ do } x := x + 1$$

s fails to be terminating, but is nondivergent.

cf.
Markov's principle: $(\neg(\forall x, \neg B\ x)) \Rightarrow \exists x, B\ x$
is a classical tautology, but is not valid constructively.

Proof sketch

Unbounded total search is nondivergent

$$\frac{\sigma\ x = n \quad \neg(B\ n) \quad \tau \models \mathit{cofinally}\ (n+1)}{\sigma :: \sigma :: \tau \models \mathit{cofinally}\ n}$$

Lemma

$\mathit{cofinally}\ 0 \models \neg\mathit{infinite}.$

Proposition

$\{x = 0\}$ while $\neg(B\ x)$ do $x := x + 1$ $\{(\text{true} ** \langle B\ x\rangle) \wedge \neg\mathit{infinite}\}$