Coinductive big-step semantics and Hoare logics for nontermination

Tarmo Uustalu, Inst of Cybernetics, Tallinn
joint work with Keiko Nakata

COST Rich Models Toolkit meeting,
Madrid, 17–18 October 2013

Motivation

- Standard big-step semantics and Hoare-style program logics do not account for nonterminating program behaviors.
- But we want to reason about nonterminating behaviors (esp in the context of reactive programming).
- Solution: Devise semantics operating on coinductively defined semantic entities (traces, resumptions) and logics for reasoning about them.
- Original inspiration: Leroy's work on coinductive big-step semantics in CompCert.

Constructivity

- An angle: We do programming language theory in a constructive setting (type theory).
  We are happy to apply non-constructive principles, where necessary, but want to notice when we do.
- This is the setting of proof assistants/dependently typed programming languages like Coq/Agda.
- It is nice to have semantics executable.
- Constructive proofs are (extract to) programs.
- In a constructive setting, indications of suboptimal designs sometimes surface earlier than in a classical setting.

This talk

- I show a trace-based big-step semantics and Hoare logic for the simple imperative language While, featuring nontermination from loops
- Could add recursive procedures or consider a (higher-order) functional language
- Important extensions: interactive input-output, shared-variable concurrency
- Here all intermediate states are tracked, generally may want to single out a class of observable event, identify weakly bisimilar traces.

Big-step semantics for nontermination

Syntax of While, states

- Statements are defined inductively by the grammar:

$$s \quad ::= \quad x := e \mid \text{skip} \mid s_0; s_1 \mid \text{if } e \text{ then } s_t \text{ else } s_f \mid \text{while } e \text{ do } s_t$$

- States σ are assignments of integers to variables names.

Trace-based big-step semantics

- We describe both converging and diverging behaviors by one single evaluation relation defined coinductively.
- The leading idea is to avoid any need to decide if a computation converges or diverges.
- This requires (at least) appreciating that computations take time.
- Cf. Leroy, Grall: two separate evaluation relations, or one single evaluation relation, but no productivity.
- We will record all intermediate states, corresponding to the idea of making all intermediate states observable.

Traces

- We define traces coinductively (as non-empty possibly infinite lists of states) by

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

- (Strong) bisimilarity is defined coinductively by

$$\frac{}{\langle \sigma \rangle \sim \langle \sigma \rangle} \qquad \frac{\tau \sim \tau_*}{\sigma :: \tau \sim \sigma :: \tau_*}$$

- We want to consider bisimilar traces as equal, so require that all predicates and functions on traces are stable under bisimilarity.
- Classically, bisimilarity is nothing but equality. Constructively, one has to be more careful...

Big-step semantics

- Evaluation relates an (initial) state to a trace and is defined coinductively:

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

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

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

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

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

Big-step semantics ctd

- Extended evaluation relates an (already accumulated) trace to a (total) trace, is also defined coinductively:

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

  (coinductive prefix closure of evaluation)
- Design choice: evaluation of an expression to assign/updating of a variable and evaluation of a guard constitute take unit time.
- Consideration: every loop always progresses, e.g., we have $(\text{while true do skip}, \sigma) \Rightarrow \sigma :: \sigma :: \ldots$.
  As a minimum, evaluating a guard to true must take unit time.
- Our choice of what takes unit time gives agreement up to bisimilarity with the natural small-step semantics.

Big-step semantics ctd

- Evaluation is stable under bisimilarity:
  • If $(s, \sigma) \Rightarrow \tau$ and $\tau \sim \tau_*$, then $(s, \sigma) \Rightarrow \tau_*$.
  (Proof by coinduction.)
- Evaluation is deterministic (up to bisimilarity):
  • If $(s, \sigma) \Rightarrow \tau$ and $(s, \sigma) \Rightarrow \tau_*$, then $\tau \sim \tau_*$.
  (Proof by coinduction.)

Big-step semantics, functional-style

- How to prove that evaluation is total, i.e., that, for any s, σ, there exists τ such that $(s, \sigma) \Rightarrow \tau$?
- Constructively, we must explicitly produce a witnessing τ from s, σ.
- This means defining evaluation and extended evaluations as functions.
- Evaluation:

$$
\begin{array}{rcll}
[\![x := e]\!]\ \sigma & = & \sigma :: \langle \sigma[x \mapsto [\![e]\!]\ \sigma] \rangle & \\
[\![\mathsf{skip}]\!]\ \sigma & = & \langle \sigma \rangle & \\
[\![s_0; s_1]\!]\ \sigma & = & [\![s_1]\!]^*\ ([\![s_0]\!]\ \sigma) & \\
[\![\mathsf{if}\ e\ \mathsf{then}\ s_t\ \mathsf{else}\ s_f]\!]\ \sigma & = & [\![s_t]\!]^*\ (\sigma :: \langle \sigma \rangle) & \sigma \models e \\
 & = & [\![s_f]\!]^*\ (\sigma :: \langle \sigma \rangle) & \sigma \not\models e \\
[\![\mathsf{while}\ e\ \mathsf{do}\ s_t]\!]\ \sigma & = & [\![\mathsf{while}\ e\ \mathsf{do}\ s_t]\!]^*\ ([\![s_t]\!]^*\ (\sigma :: \langle \sigma \rangle)) & \sigma \models e \\
 & = & \sigma :: \langle \sigma \rangle & \sigma \not\models e
\end{array}
$$

(almost structurally recursive, but not the clauses for while)

Big-step semantics, functional-style ctd

- Extension:

$$\begin{array}{lcl} k^* \ (\langle\sigma\rangle) & = & k \ \sigma \\ k^* \ (\sigma :: \tau) & = & \sigma :: (k^* \ \tau) \end{array}$$

  (guarded corecursion)

- Evaluation is total:
  $\bullet$ $(s, \sigma) \Rightarrow [\![s]\!] \ \sigma$.
  (By coinduction.)

Small-step semantics

- Single-step reduction is defined in the standard fashion inductively:

$$\frac{}{(x := e, \sigma) \to (\mathsf{skip}, \sigma[x \mapsto [\![e]\!]\ \sigma])}$$

$$\frac{}{(\mathsf{skip}, \sigma) \to \sigma} \quad \frac{(s_0, \sigma) \to \sigma'}{(s_0; s_1, \sigma) \to (s_1, \sigma')} \quad \frac{(s_0, \sigma) \to (s_0', \sigma')}{(s_0; s_1, \sigma) \to (s_0'; s_1, \sigma')}$$

$$\frac{\sigma \models e}{(\mathsf{if}\ e\ \mathsf{then}\ s_t\ \mathsf{else}\ s_f, \sigma) \to (s_t, \sigma)} \quad \frac{\sigma \not\models e}{(\mathsf{if}\ e\ \mathsf{then}\ s_t\ \mathsf{else}\ s_f, \sigma) \to (s_f, \sigma)}$$

$$\frac{\sigma \models e}{(\mathsf{while}\ e\ \mathsf{do}\ s_t, \sigma) \to (s_t; \mathsf{while}\ e\ \mathsf{do}\ s_t, \sigma)}$$

$$\frac{\sigma \not\models e}{(\mathsf{while}\ e\ \mathsf{do}\ s_t, \sigma) \to (\mathsf{skip}, \sigma)}$$

Small-step semantics ctd

- Maximal multi-step reduction is defined coinductively by:

$$\frac{(s,\sigma) \to \sigma'}{(s,\sigma) \leadsto \langle \sigma' \rangle} \qquad \frac{(s,\sigma) \to (s',\sigma') \quad (s',\sigma') \leadsto \tau}{(s,\sigma) \leadsto \sigma :: \tau}$$

- Similarly to evaluation, maximal multi-step reduction is stable under bisimilarity and deterministic up to bisimilarity.

Big-step vs small-step semantics

- Big-step semantics is sound wrt. small-step semantics:
  • If $(s, \sigma) \Rightarrow \tau$, then $(s, \sigma) \rightsquigarrow \tau$.
  (Proof by coinduction.)
- It is also complete:
  • If $(s, \sigma) \rightsquigarrow \tau$, then $(s, \sigma) \Rightarrow \tau$.
  • If $(s, \tau) \overset{*}{\rightsquigarrow} \tau'$, then $(s, \tau) \overset{*}{\Rightarrow} \tau'$.
  (Proof by mutual coinduction.)
- (Here $\rightsquigarrow^*$ is the coinductive prefix closure of $\rightsquigarrow$.)
- The following midpoint lemma is required:
  • If $(s_0; s_1, \sigma) \rightsquigarrow \tau'$, then there exists τ such that $(s_0, \sigma) \rightsquigarrow \tau$ and $(s_1, \tau) \overset{*}{\rightsquigarrow} \tau'$.
  (Proof: τ is constructed by corecursion and the two conditions are proved by coinduction.)

Hoare logic

Hoare logic

- We present a Hoare logic corresponding to the trace-based big-step semantics.
- This uses assertions on both states and traces. We don't define a fixed syntax of assertions, instead we use predicates. Trace predicates must be stable under bisimilarity.
- For trace assertions (predicates), we need some interesting connectives (operations on predicates).
- Proofs are defined inductively, as in standard Hoare logic.

Assertion connectives

- Some trace predicates:

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

$$\frac{}{\sigma :: \langle\sigma[x \mapsto [\![e]\!]\ \sigma]\rangle \models [x \mapsto e]} \qquad \frac{}{\sigma :: \langle\sigma\rangle \models \triangle}$$

$$\frac{}{\langle\sigma\rangle \models \mathit{finite}} \qquad \frac{\tau \models \mathit{finite}}{\sigma :: \tau \models \mathit{finite}} \qquad \frac{\tau \models \mathit{infinite}}{\sigma :: \tau \models \mathit{infinite}}$$

- All these predicates are stable under bisimilarity.

Assertion connectives ctd

- Chop and dagger:

$$\frac{\tau_0 \models P \quad \tau \models_{\tau_0} Q}{\tau \models P ** Q} \qquad \frac{}{\langle \sigma \rangle \models P^\dagger} \qquad \frac{\tau_0 \models P \quad \tau \models_{\tau_0} P^\dagger}{\tau \models P^\dagger}$$

where

$$\frac{\langle \sigma \rangle \models Q}{\langle \sigma \rangle \models_{\langle \sigma \rangle} Q} \qquad \frac{\sigma : \tau \models Q}{\sigma : \tau \models_{\langle \sigma \rangle} Q} \qquad \frac{\tau \models_{\tau_0} Q}{\sigma : \tau \models_{\sigma : \tau_0} Q}$$

- Cf. interval temporal logic, B. Mosztowski
- If P, Q are stable under bisimilarity, then so is $P ** Q$. If P is stable under bisimilarity, so is $P^\dagger$.
- If τ is infinite and $\tau \models P$, then $\tau \models P ** Q$ for any Q!

Hoare proofs

- Proofs are defined inductively by the rules

$$\frac{}{\{U\}\ x := e\ \{\langle U\rangle ** [x \mapsto e]\}}$$

$$\frac{}{\{U\}\ \mathrm{skip}\ \{\langle U\rangle\}} \qquad \frac{\{U\}\ s_0\ \{P ** \langle V\rangle\} \quad \{V\}\ s_1\ \{Q\}}{\{U\}\ s_0; s_1\ \{P ** Q\}}$$

$$\frac{\{e \wedge U\}\ s_t\ \{P\} \quad \{\neg e \wedge U\}\ s_f\ \{P\}}{\{U\}\ \mathrm{if}\ e\ \mathrm{then}\ s_t\ \mathrm{else}\ s_f\ \{\triangle ** P\}}$$

$$\frac{U \models I \quad \{e \wedge I\}\ s_t\ \{P ** \langle I\rangle\}}{\{U\}\ \mathrm{while}\ e\ \mathrm{do}\ s_t\ \{(\triangle ** P)^\dagger ** \triangle ** \langle \neg e\rangle\}}$$

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

Soundness

- The Hoare logic is sound.

  - If $\{U\}\ s\ \{P\}$, then $\sigma \models U$ and $(s, \sigma) \Rightarrow \tau$ imply $\tau \models P$.
- (By induction on $\{U\}\ s\ \{P\}$, subordinate coinduction in several cases.)

Completeness

- The Hoare logic is also complete.

  • If, for any σ, τ, $\sigma \models U$ and $(s, \sigma) \Rightarrow \tau$ imply $\tau \models P$, then $\{U\}\ s\ \{P\}$.

- To prove this, one defines for any s, U, a trace predicate $sp(s, U)$ (the strongest postcondition), by structural recursion on s.
- Now completeness follows from these lemmata:
  • $\{U\}\ s\ \{sp(s, U)\}$. (By induction on s).
  • If $\tau \models sp(s, U)$, then $(s, hd\ \tau) \Rightarrow \tau$. (By induction on s.)
  The latter gives as an immediate corollary:
  • If, for all σ, τ, $\sigma \models U$ and $(s, \sigma) \Rightarrow \tau$ imply $\tau \models P$, then $sp(s, U) \models P$.

Strongest postconditions

- Strongest postconditions:

$$
\begin{array}{lcl}
sp(x := e, U) & = & \langle U \rangle ** [x \mapsto e] \\
sp(\text{skip}, U) & = & \langle U \rangle \\
sp(s_0; s_1, U) & = & sp(s_0, U) ** sp(s_1, Last\ (sp(s_0, U))) \\
sp(\text{if } e \text{ then } s_t \text{ else } s_f, U) & = & \langle U \rangle ** \triangle ** (sp(s_t, e \wedge U) \vee sp(s_f, \neg e \wedge U)) \\
sp(\text{while } e \text{ do } s_t, U) & = & \langle U \rangle ** (\triangle ** sp(s_t, e \wedge Inv(e, s_t, U)))^{\dagger} ** \triangle ** \langle \neg e \rangle
\end{array}
$$

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

$$
\frac{\sigma \models U}{\sigma \models Inv(e, s, U)} \qquad \frac{\sigma \models Last(sp(s, e \wedge Inv(e, s, U)))}{\sigma \models Inv(e, s, U)}
$$

Embedding of standard Hoare logic

- The standard Hoare logic embeds into the Hoare logic for trace-based semantics.
  • If $\{U\}\ s\ \{Z\}$ in the standard (partial correctness) Hoare logic, then $\{U\}\ s\ \{\mathit{true} ** \langle Z\rangle\}$.
  (Could go via soundness and completeness. But there is a direct proof by induction.)

- Similarly, the total-correctness Hoare logic also embeds into our logic.
  • If $\{U\}\ s\ \{Z\}$ in the total correctness Hoare logic, then $\{U\}\ s\ \{\mathit{finite} ** \langle Z\rangle\}$.