Iris: Higher-Order Concurrent Separation Logic

Lecture 3: Basic Separation Logic: Hoare Triples

Lars Birkedal

Aarhus University, Denmark

November 10, 2017

Overview

Earlier:

- Operational Semantics of $\lambda_{\mathrm{ref,conc}}$
  - e, $(h, e) \rightsquigarrow (h, e')$, and $(h, \mathcal{E}) \to (h', \mathcal{E}')$
- Basic Logic of Resources
  - $l \hookrightarrow v$, $P * Q$, $P \mathrel{-\!\!*} Q$, $\Gamma \mid P \vdash Q$

Today:

- Basic Separation Logic: Hoare Triples
  - $\{P\}\, e\, \{v.Q\} : \mathrm{Prop}$

Hoare Triples

$$\frac{\Gamma \vdash P : \mathsf{Prop} \qquad \Gamma \vdash e : \mathit{Exp} \qquad \Gamma \vdash \Phi : \mathit{Val} \to \mathsf{Prop}}{\Gamma \vdash \{P\}\, e\, \{\Phi\} : \mathsf{Prop}}$$

Intuition

- $\{P\}\, e\, \{\Phi\}$ holds if, when we run the program e in a heap h satisfying P, then the computation does not get stuck and, moreover, if it terminates with a value v and a heap h', then h' satisfies $\Phi(v)$.
- Φ has two purposes: describes the value v (*e.g.*, $v = 3$) and the resources after execution (*e.g.*, $x \hookrightarrow 15$).
- Note that Φ is a function – we often write Φ as $v.\, Q$ instead of $\lambda v.\, Q$.

Examples

$$\{l \hookrightarrow 5\}\, l \leftarrow !\, l + 1\, \{v.v = () \wedge l \hookrightarrow 6\}$$

$$\{\ell_1 \hookrightarrow v_1 * \ell_2 \hookrightarrow v_2\}\, \mathsf{swap}\ \ell_1\, \ell_2\, \{v.v = () \wedge \ell_1 \hookrightarrow v_2 * \ell_2 \hookrightarrow v_1\}.$$

Hoare Triples

More intuition

- Precondition P describes the resources necessary to run e safely (recall "does not get stuck" in the intuitive reading above).
- In our operational semantics, memory errors, *e.g.*, trying to dereference a location that has not been allocated, are modelled by the computation getting stuck.
- So if e satisfies a Hoare triple, then its computation will not lead to any memory errors.
- Precondition P describes the resources needed for e to run safely: we sometimes say that P includes the *footprint* of e.
- (Later on, not all resources needed to execute e will need to be in the precondition — resources shared among different threads will be in invariants, and only resources owned by e's thread will in the precondition.)

Frame Rule

$$\textsc{Ht-frame}\quad \frac{S \vdash \{P\}\, e\, \{v.Q\}}{S \vdash \{P * R\}\, e\, \{v.Q * R\}}$$

- Intuitively sound because of the footprint reading of triples
- Note that the frame R is maintained unchanged from precondition to postcondition.
- We do not have to explicitly say that e does not modify other resources not in its precondition!
- Very important!
- We will use this rule all the time.

Frame Rule

Example

- Consider the specification for swap:

$$\{\ell_1 \hookrightarrow v_1 * \ell_2 \hookrightarrow v_2\}\,\text{swap}\ \ell_1\,\ell_2\,\{v.v = () \wedge \ell_1 \hookrightarrow v_2 * \ell_2 \hookrightarrow v_1\}.$$

- What if we want to apply this function somewhere, where we have more resources around ? For instance $\ell_3 \hookrightarrow 3$. Then we use the frame rule, with frame $R = \ell_3 \hookrightarrow 3$, to derive

$$\{\ell_1 \hookrightarrow v_1 * \ell_2 \hookrightarrow v_2 * \ell_3 \hookrightarrow 3\}\,\text{swap}\ \ell_1\,\ell_2\,\{v.v = () \wedge \ell_1 \hookrightarrow v_2 * \ell_2 \hookrightarrow v_1 * \ell_3 \hookrightarrow 3\}.$$

Value Rule

$$\frac{\textsc{Ht-ret} \qquad w \text{ is a value}}{S \vdash \{\mathsf{True}\}\, w\, \{v.\, v = w\}}$$

Rule for Binary Operators

Basic rules are given for values, *e.g.*,

$$\textsc{Ht-binop}\quad \frac{v_1 \text{ and } v_2 \text{ are values}}{S \vdash \{\mathrm{True}\}\, v_1 \circledcirc v_2 \,\{v.\, v = v_1 \circledcirc v_2\}}$$

Here the latter $\circledcirc$ is the mathematical operation corresponding to the syntactic operator.

Bind Rule

To verify larger expressions we use the $\textsc{Ht-bind}$ rule:

$$\frac{\textsc{Ht-bind}\quad E \text{ is an eval. context} \qquad S \vdash \{P\}\, e\, \{v.\, Q\} \qquad S \vdash \forall v.\, \{Q\}\, E[v]\, \{w.\, R\}}{S \vdash \{P\}\, E[e]\, \{w.\, R\}}$$

- Exercise: Use $\textsc{Ht-bind}$ to show $\{\text{True}\}\, 3 + 4 + 5\, \{v.v = 12\}$.

Persistent Propositions

- Intuition: persistent propositions are propositions that do not rely on resources, *i.e.*, either they hold for all resources or none.

$$P \wedge Q \vdash P * Q \qquad \text{if } P \text{ is persistent.}$$

- Persistent propositions may be moved in and out of preconditions:

$$\textsc{Ht-Eq}\;\; \frac{S \wedge t =_\tau t' \vdash \{P\}\, e\, \{v.\, Q\}}{S \vdash \{P \wedge t =_\tau t'\}\, e\, \{v.\, Q\}} \qquad \textsc{Ht-Ht}\;\; \frac{S \wedge \{P_1\}\, e_1\, \{v.\, Q_1\} \vdash \{P_2\}\, e_2\, \{v.\, Q_2\}}{S \vdash \{P_2 \wedge \{P_1\}\, e_1\, \{v.\, Q_1\}\}\, e_2\, \{v.\, Q_2\}}$$

- For now it suffices to know that persistence is preserved by $\forall$ and $\wedge$ — we will see a general treatment later.

Example of HT-HT

$$\frac{\{x \hookrightarrow 5\}\,\mathsf{inc}\,x\,\{v.v = () \wedge x \hookrightarrow 6\} \vdash \{x \hookrightarrow 5\}\,\mathsf{inc}\,x\,\{v.v = () \wedge x \hookrightarrow 6\}}{\{x \hookrightarrow 5 \wedge \{x \hookrightarrow 5\}\,\mathsf{inc}\,x\,\{v.v = () \wedge x \hookrightarrow 6\}\}\,\mathsf{inc}\,x\,\{v.v = () \wedge x \hookrightarrow 6\}}$$

Consequence Rule

$$\textsc{Ht-csq}\quad \frac{S \text{ persistent} \qquad S \vdash P \Rightarrow P' \qquad S \vdash \{P'\}\, e\, \{v.\, Q'\} \qquad S \vdash \forall u.\, Q'[u/v] \Rightarrow Q[u/v]}{S \vdash \{P\}\, e\, \{v.\, Q\}}$$

Remark: S is usually a conjunction of equalities and universally quantified Hoare triples, so is usually persistent.

Load Rule

$$\frac{\textsc{Ht-load}}{S \vdash \{\ell \hookrightarrow u\}\; !\,\ell\; \{v.\, v = u \wedge \ell \hookrightarrow u\}}$$

- Intuitively sound because ...

Alloc Rule

$$\dfrac{\textsc{Ht-alloc}}{S \vdash \{\mathsf{True}\}\, \mathsf{ref}(u)\, \{v. \exists \ell.\, v = \ell \land \ell \hookrightarrow u\}}$$

- Intuitively sound because ...

Store Rule

$$\frac{\textsc{Ht-store}}{S \vdash \{\ell \hookrightarrow -\}\, \ell \leftarrow w\, \{v. v = () \wedge \ell \hookrightarrow w\}}$$

- $\ell \hookrightarrow -$ shorthand for $\exists u.\, \ell \hookrightarrow u$
- Intuitively sound because ...

Rules for Conditionals

$$\textsc{Ht-If-True}\quad \frac{\{P * v = \mathsf{true}\}\; e_2\; \{u.Q\}}{\{P * v = \mathsf{true}\}\; \mathsf{if}\; v\; \mathsf{then}\; e_2\; \mathsf{else}\; e_3\; \{u.Q\}}$$

$$\textsc{Ht-If-False}\quad \frac{\{P * v = \mathsf{false}\}\; e_3\; \{u.Q\}}{\{P * v = \mathsf{false}\}\; \mathsf{if}\; v\; \mathsf{then}\; e_2\; \mathsf{else}\; e_3\; \{u.Q\}}$$

$$\textsc{Ht-If}\quad \frac{\{P * v = \mathsf{true}\}\; e_1\; \{u.Q\} \qquad \{P * v = \mathsf{false}\}\; e_3\; \{u.Q\}}{\{P\}\; \mathsf{if}\; v\; \mathsf{then}\; e_2\; \mathsf{else}\; e_3\; \{u.Q\}}$$

Rules for Products and Sums

$$\textsc{Proj}\quad \frac{}{S \vdash \{\mathsf{True}\}\ \pi_i\,(v_1, v_2)\ \{v.\,v = v_i\}}$$

$$\textsc{Match}\quad \frac{S \vdash \{P\}\ e_i\,[u/x_i]\ \{v.\,Q\}}{S \vdash \{P\}\ \mathsf{match}\ \mathsf{inj}_i\ u\ \mathsf{with}\ x_1 \Rightarrow e_1 \mid x_2 \Rightarrow e_2\ \mathsf{end}\ \{v.\,Q\}}$$

Recursion Rule

$$\frac{\Gamma, f : \mathit{Val} \mid S \wedge \forall y.\, \forall v.\, \{P\}\, f\ v\, \{u.Q\} \vdash \forall y.\, \forall v.\, \{P\}\, e[v/x]\, \{u.Q\}}{\Gamma \mid S \vdash \forall y.\, \forall v.\, \{P\}\, (\mathbf{rec}\, f(x) = e) v\, \{u.Q\}}\ \textsc{Ht-Rec}$$

- Here y is a "logical" variable, which may be used in P and Q to relate pre and postconditions. Example:
  - $\forall y : \mathbb{N}. \forall x. \{x = y\}\, \mathsf{double}\, x\, \{v.\ v =_{\mathit{Val}} 2 \times y\}$
- When reasoning about the body, we get to assume that f satisfies the triple we are about to prove.
- Intuitively sound by induction on reduction steps.

Exercise (jointly, on the board)

- Specify and prove a functional implementation of factorial.

Factorial

Implementation

- $\mathsf{rec}\ \mathsf{fac}(n) = \mathsf{if}\ n = 0\ \mathsf{then}\ 1\ \mathsf{else}\ n * \mathsf{fac}(n-1)$

Specification

- $\forall n : \mathbb{N}. \{\mathrm{True}\}\ \mathsf{fac}\ n\ \{v.\ v =_{Val} n!\}$

Proof

- Use the recursion rule!