Iris: Higher-Order Concurrent Separation Logic

Lecture 12: The Authoritative Resource Algebra: Concurrent Counter Modules

Lars Birkedal

Aarhus University, Denmark

November 30, 2017

Overview

Earlier:

- Operational Semantics of $\lambda_{\text{ref,conc}}$
  - e, $(h, e) \leadsto (h, e')$, and $(h, \mathcal{E}) \to (h', \mathcal{E}')$
- Basic Logic of Resources
  - $l \hookrightarrow v$, $P * Q$, $P \twoheadrightarrow Q$, $\Gamma \mid P \vdash Q$
- Basic Separation Logic
  - $\{P\}\, e\, \{v.Q\}$: Prop, isList l xs, ADTs, foldr
- Later ($\triangleright$) and Persistent ($\Box$) Modalities.
- Concurrency Intro, Invariants and Ghost State
- CAS and Spin Locks.

Today:

- Proof patterns for concurrency
- Key Points:
  - Authoritative Resource Algebra.
  - Fractions to track concurrent ownership.

A Recurring Specification and Proof Pattern

- Wish to consider situation where several threads operate on shared state.
- Each thread has a *partial view* or *fragmental view* of the shared state.
- There is an invariant governing the shared state.
- The invariant keeps track of what the actual state is, hence it tracks the *authoritative view* of the shared state.

Example: Counter Module

- Counter module with three methods:
  - newCounter for creating a fresh counter,
  - incr for increasing the value of the counter,
  - read for reading the current value of the counter.
- Abstract predicate isCounter(v, n): v is a counter whose current value is n.
- isCounter(v, n) should be persistent, so different threads can access the counter simultaneously.
- Hence isCounter(v, n) cannot state that n is *exactly* the value of the counter, but only its lower bound.

Counter Implementation

- The newCounter method creates the counter: a location containing the counter value.

$$\mathsf{newCounter}() = \mathsf{ref}(0)$$

- The incr method increases the value of the counter by 1. Since $\ell \leftarrow !\,\ell + 1$ is not an atomic operation we use a cas loop:

$$\begin{aligned}\mathsf{rec}\,\mathsf{incr}(\ell) = {} & \mathsf{let}\, n = !\,\ell \,\mathsf{in} \\ & \mathsf{let}\, m = n + 1 \,\mathsf{in} \\ & \mathsf{if}\,\mathsf{cas}(\ell, n, m)\,\mathsf{then}\,()\,\mathsf{else}\,\mathsf{incr}\,\ell\end{aligned}$$

- The read method simply reads the value

$$\mathsf{read}\,\ell = !\,\ell.$$

Authoritative and Fragmental Views

- We will use an invariant to keep track of the shared state of the module, the value of the counter.
- The invariant will have the *authoritative view* of the value of the counter, a ghost assertion:

  $$\boxed{\bullet\, m}^{\gamma}$$

  Intuitively, this is the correct, true, value of the counter.
- Each thread will have a *fragmental view* of the value of the counter, captured by a ghost assertion:

  $$\boxed{\circ\, n}^{\gamma}$$

  Intuitively, this is a lower bound of the correct, true, value of the counter.

Authoritative and Fragmental Views

- We will use an invariant to keep track of the shared state of the module, the value of the counter.
- The invariant will have the *authoritative view* of the value of the counter, a ghost assertion:

  $$\boxed{\bullet\, m}^{\gamma}$$

  Intuitively, this is the correct, true, value of the counter.
- Each thread will have a *fragmental view* of the value of the counter, captured by a ghost assertion:

  $$\boxed{\circ\, n}^{\gamma}$$

  Intuitively, this is a lower bound of the correct, true, value of the counter.
- Define abstract predicate by

$$\mathrm{isCounter}(\ell, n, \gamma) = \boxed{\circ\, n}^{\gamma} * \exists \iota.\, \boxed{\exists m.\, \ell \mapsto m * \boxed{\bullet\, m}^{\gamma}}^{\iota}$$

RA requirements

$$|\circ n| = \circ n \tag{1}$$

$$\bullet m \cdot \circ n \in \mathcal{V} \Rightarrow m \geq n \tag{2}$$

$$\bullet m \cdot \circ n \rightsquigarrow \bullet (m+1) \cdot \circ (n+1) \tag{3}$$

1. a fragmental view should be duplicable (several threads may share the same fragmental view, *i.e.*, several threads may agree that the lower bound of counter is n, say)
2. the fragmental view is a lower bound of the true value
3. if we own both the authoriative view and a fragmental view, then we may update them (so we can only update a fragmental view, if we also update the authoritative view!)

RA definition

- Carrier: $\mathcal{M} = \mathbb{N}_{\bot,\top} \times \mathbb{N}$ where $\mathbb{N}_{\bot,\top}$ is the naturals with two additional elements $\bot$ and $\top$.
  - Idea: for $m, n \in \mathbb{N}$, write $\bullet\, m$ for $(m, 0)$ and $\circ\, n$ for $(\bot, n)$.
- Operation:

$$(x, n) \cdot (y, m) = \begin{cases} (y, \max(n, m)) & \text{if } x = \bot \\ (x, \max(n, m)) & \text{if } y = \bot \\ (\top, \max(n, m)) & \text{otherwise} \end{cases}$$

- Unit: $(\bot, 0)$.
- Validity

$$\mathcal{V} = \{(x, n) \mid x = \bot \lor x \in \mathbb{N} \land x \geq n\}\,.$$

- Core

$$|(x, n)| = (\bot, n).$$

- $(\mathcal{M}, \mathcal{V}, |\cdot|)$ is a unital resource algebra.

RA definition

- For $m, n \in \mathbb{N}$, write $\bullet\, m$ for $(m, 0)$ and $\circ\, n$ for $(\perp, n)$.
- Then the required properties hold.

Checking required properties: example

Let us check $\bullet\, m \cdot \circ\, n \rightsquigarrow \bullet\,(m+1) \cdot \circ\,(n+1)$:

- First, recall that
  - $\bullet\, m \cdot \circ\, n = (m, 0) \cdot (\bot, n) = (m, n)$, and
  - $\bullet\,(m+1) \cdot \circ\,(n+1) = (m+1, 0) \cdot (\bot, n+1) = (m+1, n+1)$.
- TS, for all (x, y),

$$(m, n) \cdot (x, y) \in \mathcal{V} \Rightarrow (m+1, n+1) \cdot (x, y) \in \mathcal{V}.$$

- So suppose $(m, n) \cdot (x, y) \in \mathcal{V}$. Then $x = \bot$, and $(m, n) \cdot (x, y) = (m, \max(n, y))$ and $\max(n, y) \leq m$.
- But then also $\max(n+1, y) \leq m+1$ and hence $(m+1, \max(n+1, y)) = (m+1, n+1) \cdot (x, y) \in \mathcal{V}$, as required.

Counter Specification and Client

Exercise: Show the following specifications:

$$\{\mathsf{True}\}\ \mathsf{newCounter}()\ \{u.\exists\gamma.\ \mathsf{isCounter}(u, 0, \gamma)\}$$
$$\forall\gamma.\,\forall v.\,\forall n.\,\{\mathsf{isCounter}(v, n, \gamma)\}\ \mathsf{read}\ v\ \{u.u \geq n\}$$
$$\forall\gamma.\,\forall v.\,\forall n.\,\{\mathsf{isCounter}(v, n, \gamma)\}\ \mathsf{incr}\ v\ \{u.u = () * \mathsf{isCounter}(v, n+1, \gamma)\}$$

Let e be the program

$$\mathsf{let}\ c = \mathsf{newCounter}()\ \mathsf{in}\ (\mathsf{incr}\ c \,||\, \mathsf{incr}\ c);\ \mathsf{read}\ c.$$

Show the following specification for e.

$$\{\mathsf{True}\}\ e\ \{v.v \geq 2\}.$$

A More Precise Spec ?

- For the example program e above, we know operationally that the final value will be 2.
- However, we cannot prove that with out spec, since isCounter is freely duplicable:
  - we do not track whether other threads are using the counter.
- Now we will show how to use *fractions* to keep track of concurrent ownership.

Fractions to track concurrent ownership of counter

- Add fraction q to the abstract isCounter predicate:
  - Intuition: If a thread has ownership of $\mathsf{isCounter}(\ell, n, \gamma, q)$, then
  - the contribution of this thread to the actual counter value is n, and
  - if $q = 1$, then this thread is the sole owner, otherwise ($q < 1$) we have fragmental ownership.
- Specification: (note two specs for read):

$$\{\mathsf{True}\}\ \mathsf{newCounter}()\ \{u.\exists\gamma.\ \mathsf{isCounter}(u, 0, \gamma, 1)\}$$
$$\forall p. \forall\gamma. \forall v. \forall n.\ \{\mathsf{isCounter}(v, n, \gamma, p)\}\ \mathsf{read}\ v\ \{u.u \geq n\}$$
$$\forall\gamma. \forall v. \forall n.\ \{\mathsf{isCounter}(v, n, \gamma, 1)\}\ \mathsf{read}\ v\ \{u.u = n\}$$
$$\forall p. \forall\gamma. \forall v. \forall n.\ \{\mathsf{isCounter}(v, n, \gamma, p)\}\ \mathsf{incr}\ v\ \{u.u = () * \mathsf{isCounter}(v, n+1, \gamma, p)\}$$

- isCounter is not persistent anymore; instead we have:

$$\mathsf{isCounter}(\ell, n+k, \gamma, p+q) \dashv\vdash \mathsf{isCounter}(\ell, n, \gamma, p) * \mathsf{isCounter}(\ell, k, \gamma, q).$$

Authoritative Resource Algebra Construction $\textsc{Auth}(\mathcal{M})$

- Given a *unital* RA $(\mathcal{M}, \varepsilon, \mathcal{V}, |\cdot|)$, let $\textsc{Auth}(\mathcal{M})$ be RA with
  - Carrier: $\mathcal{M}_{\bot,\top} \times \mathcal{M}$
  - Operation:

$$(x, a) \cdot (y, b) = \begin{cases} (y, a \cdot b) & \text{if } x = \bot \\ (x, a \cdot b) & \text{if } y = \bot \\ (\top, a \cdot b) & \text{otherwise} \end{cases}$$

  - Core:

$$|(x, a)|_{\textsc{Auth}(\mathcal{M})} = (\bot, |a|)$$

  - Valid elements:

$$\mathcal{V}_{\textsc{Auth}(\mathcal{M})} = \left\{ (x, a) \;\middle|\; x = \bot \land a \in \mathcal{V} \lor x \in \mathcal{M} \land x \in \mathcal{V} \land a \preccurlyeq x \right\}$$

- We write $\bullet\, m$ for (m, ε) and $\circ\, n$ for $(\bot, n)$.

Properties of $\textsc{Auth}(\mathcal{M})$

- $\textsc{Auth}(\mathcal{M})$ is unital with unit $(\bot, \varepsilon)$, where ε is the unit of $\mathcal{M}$
- $\bullet\, x \cdot \bullet\, y \notin \mathcal{V}_{\textsc{Auth}(\mathcal{M})}$ for any x and y
- $\circ\, x \cdot \circ\, y = \circ\,(x \cdot y)$
- $\bullet\, x \cdot \circ\, y \in \mathcal{V} \Rightarrow y \preccurlyeq x$
- if $x \cdot z$ is valid in $\mathcal{M}$ then

  $$\bullet\, x \cdot \circ\, y \rightsquigarrow \bullet\,(x \cdot z) \cdot \circ\,(y \cdot z)$$

  in $\textsc{Auth}(\mathcal{M})$

(Exercise!)

- Remark: The RA we used earlier for the counter is $\textsc{Auth}(\mathbb{N}_{\mathsf{max}})$, where $\mathbb{N}_{\mathsf{max}}$ is the RA with carrier the natural number and operation the maximum, core the identity function and all elements valid.

Verifying the more precise spec

- New def'n of representation predicate:

$$\mathrm{isCounter}(\ell, n, \gamma, p) = \boxed{\circ\,(p, n)}^{\gamma} * \exists \iota.\, \boxed{\exists m.\, \ell \mapsto m * \boxed{\bullet\,(1, m)}^{\gamma}}^{\iota}.$$

- Idea: invariant stores the exact value of the counter, hence the fraction is 1.
- Fragment $\boxed{\circ\,(p, n)}^{\gamma}$ connects the actual value of the counter to the value known to a particular thread.
- Thus, to be able to read the exact value of the counter when p is 1 we need the property that if $\bullet\,(1, m) \cdot \circ\,(1, n)$ is valid then $n = m$.
- Further,need that if $\bullet\,(1, m) \cdot \circ\,(p, n)$ is valid then $m \geq n$.
- Finally, wish
  $\mathrm{isCounter}(\ell, n + k, \gamma, p + q) \dashv\vdash \mathrm{isCounter}(\ell, n, \gamma, p) * \mathrm{isCounter}(\ell, k, \gamma, q)$.

Verifying the more precise spec: choice of RA

- Achieve the above by using $\textsc{Auth}((\mathbb{Q}_{01} \times \mathbb{N})_?)$, where
  - $\mathbb{Q}_{01}$ is the RA of fractions.
  - $\mathbb{N}$ is the resource algebra of natural numbers with *addition* as the operation, and every element is valid,
  - $(\mathbb{Q}_{01} \times \mathbb{N})_?$ is the option RA on the product of the two previous ones.
- Properties:
  - $\circ (p, n) \cdot \circ (q, m) = \circ (p + q, n + m)$
  - if $\bullet (1, m) \cdot \circ (p, n)$ is valid then $n \leq m$ and $p \leq 1$
  - if $\bullet (1, m) \cdot \circ (1, n)$ is valid then $n = m$
  - $\bullet (1, m) \cdot \circ (p, n) \rightsquigarrow \bullet (1, m + 1) \cdot \circ (p, n + 1)$.

Verifying the more precise spec

With isCounter defined as shown above, we get

$$\text{isCounter}(\ell, n+k, \gamma, p+q) \dashv\vdash \text{isCounter}(\ell, n, \gamma, p) * \text{isCounter}(\ell, k, \gamma, q).$$

and

$$\{\text{True}\}\ \text{newCounter}()\ \{u.\exists\gamma.\ \text{isCounter}(u, 0, \gamma, 1)\}$$
$$\forall p.\,\forall\gamma.\,\forall v.\,\forall n.\ \{\text{isCounter}(v, n, \gamma, p)\}\ \text{read}\ v\ \{u.u \geq n\}$$
$$\forall\gamma.\,\forall v.\,\forall n.\ \{\text{isCounter}(v, n, \gamma, 1)\}\ \text{read}\ v\ \{u.u = n\}$$
$$\forall p.\,\forall\gamma.\,\forall v.\,\forall n.\ \{\text{isCounter}(v, n, \gamma, p)\}\ \text{incr}\ v\ \{u.u = () * \text{isCounter}(v, n+1, \gamma, p)\}$$

Let e be the program

$$\textbf{let}\ c = \text{newCounter}()\ \textbf{in}\ (\text{incr}\ c \,||\, \text{incr}\ c);\ \text{read}\ c.$$

Now one can use the above spec to show:

$$\{\text{True}\}\ e\ \{v.v = 2\}.$$