Using Iris for Program Verification

Abel Nieto

March 18, 2020

1 Monotone Counter

▷ **Coq code:** `https://github.com/abeln/iris-practice/blob/master/counter.v` ◁

This is the counter example from Section 7.7 of the notes. The module exports three methods:

- `newCounter` creates a new counter. Counters are represented simply with an int reference.

  ```
  Definition newCounter : val := \lambda: <>, ref #0.
  ```

- `read` returns the value currently stored in a counter.

  ```
  Definition read : val := \lambda: "c", !"c".
  ```

- `incr` takes a counter, increments it, and returns unit.

  ```
  Definition incr : val :=
    rec: "incr" "c" :=
      let: "n" := !"c" in
      let: "m" := #1 + "n" in
      if: CAS "c" "n" "m" then #() else "incr" "c".
  ```

The client for the counter is a program that instantiates a counter, spawns two threads each incrementing the counter, and then reads the value off the counter:

```
Definition client : val :=
 \lambda: <>,
   let: "c" := newCounter #() in
   ((incr "c") ||| (incr "c")) ;;
   read "c".
```

1.1 Specs

We can prove two different specs for the code above:

- Authoritative RA

  In the first spec, our resource algebra is $\textsc{Auth}(\mathbb{N})$. Elements of this RA are of the form either $\bullet n$ (authoritative) or $\circ n$ (non-authoritative), where $n \in \mathbb{N}$.

  The important properties of this RA (for the counter example) are:

  - $\bullet 0 \cdot \circ 0 \in \mathcal{V}$
  - $\bullet m \cdot \circ n \in \mathcal{V}$ implies $m \geq n$
  - $\bullet m \cdot \circ n \rightsquigarrow \bullet(m+1) \cdot \circ(n+1)$
  - $|\circ n| = \circ n$

  Then we can define our predicate for the counter:

  $$\text{isCounter}(l, n, \gamma) = \ulcorner \circ n \urcorner^{\gamma} * \exists s. \boxed{\exists m. l \hookrightarrow m \land \ulcorner \bullet m \urcorner^{\gamma}}^{s}$$

  This predicate is persistent because the invariant component is persistent and the core of $\circ n$ is $\circ n$ itself (via the Persistently-core rule).

  We then prove the following specs for the counter methods:

  - $\{\top\}$ newCounter () $\{l.\exists\gamma.\text{isCounter}(l, 0, \gamma)\}$
  - $\{\text{isCounter}(l, n, \gamma)\}$ read l $\{v.v \geq n\}$. Notice how we don't have to give the counter back, because it's persistent.
  - $\{\text{isCounter}(l, n, \gamma)\}$ incr l $\{\text{isCounter}(l, n+1, \gamma)\}$

  We can also prove the client spec:

  $$\{\top\}\text{client }()\{n.n \geq 1\}$$

- Authoritative RA + Fractions