Unary logical relation with mutable store in Iris

Jacques-Henri Jourdan

November 19th, 2024

Definitions in `logical_relation.v`.

1 From semantic types to adequacy

2 Simple types

3 References

4 ∀ quantifier

Integers, Booleans, sums

Definitions of the semantic type in `int.v`, `bool.v` and `sum.v`.

Pairs

Definitions and proofs in `pair.v`.

Arrows

Definitions and proofs in `arrow.v`.

1 From semantic types to adequacy

2 Simple types

3 References

4 $\forall$ quantifier

Logical relation with references

First definition attempt

To understand the problem, we go back to the on-paper formalism.

$$\mathcal{V}(\texttt{ref}(\tau)) = \{\ell \in \mathsf{Loc} \mid ???\}$$

Logical relation with references

First definition attempt

To understand the problem, we go back to the on-paper formalism.

$$\mathcal{V}(\texttt{ref}(\tau)) = \{\ell \in \mathsf{Loc} \mid ???\}$$

We need to parameterize the logical relation with a store typing M, giving semantic types to locations. Roughly, we want:

$$\mathcal{V}(\texttt{ref}(\tau)) = \lambda M.\ \{\ell \in \mathsf{Loc} \mid M(\ell) = \mathcal{V}(\tau)\}$$

Logical relation with references

The cardinality problem

In which set would $\mathcal{V}(\tau)$ live?

Let TypeSem be the set of semantic types: $\mathcal{V}(\tau) \in \mathsf{TypeSem}$.
Let StoreType be the set of store typing: $M \in \mathsf{StoreType}$.

We have:

$$\begin{aligned} \mathsf{TypeSem} &= \mathsf{StoreType} \to \mathcal{P}(\mathsf{Val}) \\ \mathsf{StoreType} &= \mathsf{Loc} \to_{fin} \mathsf{TypeSem} \end{aligned}$$

These equations are contradictory for cardinality reasons!
We cannot define the logical relation, because the set it belongs to does not exists!

Step indexing

Idea: for each $k \in \mathbb{N}$, we build a logical relation $\mathcal{V}_k$ to prove safety only for the next k steps.

Definition of the relation for $\texttt{ref}(\tau)$:

$$\begin{aligned} \mathcal{V}_{k+1}(\texttt{ref}(\tau)) &= \lambda M.\, \{\ell \in \mathsf{Loc} \mid M(\ell) = \mathcal{V}_k(\tau)\} \\ \mathcal{V}_0(\texttt{ref}(\tau)) &= \lambda M.\, \mathsf{Loc} \end{aligned}$$

This solves the cardinality problem. We have $\mathcal{V}_0(\texttt{ref}(\tau)) \in \mathsf{TypeSem}_k$ with:

$$\begin{aligned} \mathsf{TypeSem}_k &= \mathsf{StoreType}_k \to \mathcal{P}(\mathsf{Val}) \\ \mathsf{StoreType}_0 &= \mathsf{Unit} \\ \mathsf{StoreType}_{k+1} &= \mathsf{Loc} \to_{fin} \mathsf{TypeSem}_k \end{aligned}$$

Step indexing in Iris

Iris uses the step indexing technique, but step indices are hidden to the user.

Any Iris assertion P is in fact a collection of assertions $P_0, \ldots, P_k, \ldots$

The "later modality" $\triangleright$ shifts an assertion by one step:

$$(\triangleright P)_0 = \top \qquad (\triangleright P)_1 = P_0 \qquad \ldots \qquad (\triangleright P)_k = P_{k-1}$$

Note: for more detailed information on how step indexing and the $\triangleright$ modality work, we refer to Xavier Leroy course et Collège de France : "Step carefully: step-indexing techniques" (2019-01-09).

Let's go back in Coq

Definitions and proofs in `ref.v`.

1 From semantic types to adequacy

2 Simple types

3 References

4 $\forall$ quantifier

$\forall$ quantifier

Definitions and proofs in `forall.v`.