The Iris 2.0 Documentation

August 24, 2016

Contents

1 Algebraic Structures 2
  1.1 COFE ................................................................. 2
  1.2 RA ................................................................. 3
  1.3 CMRA .............................................................. 4

2 COFE constructions 6
  2.1 Next (type-level later) ............................................. 6
  2.2 Uniform Predicates ................................................ 6

3 RA and CMRA constructions 7
  3.1 Product ............................................................ 7
  3.2 Sum ................................................................. 7
  3.3 Finite partial function ............................................ 7
  3.4 Agreement .......................................................... 8
  3.5 Exclusive CMRA ................................................... 8
  3.6 STS with tokens ................................................... 9

4 Language 10
  4.1 Concurrent language ................................................ 10

5 Logic 11
  5.1 Grammar .......................................................... 11
  5.2 Types ............................................................... 12
  5.3 Proof rules ........................................................ 12
  5.4 Adequacy ............................................................ 15

6 Model and semantics 16
  6.1 Generic model of base logic ...................................... 16
  6.2 Iris model .......................................................... 16

7 Derived proof rules and other constructions 20
  7.1 Base logic .......................................................... 20
  7.2 Program logic ....................................................... 20
  7.3 Global functor and ghost ownership ............................ 22
  7.4 Invariant identifier namespaces .................................. 22
1 Algebraic Structures

1.1 COFE

The model of Iris lives in the category of Complete Ordered Families of Equivalences (COFEs). This definition varies slightly from the original one in [2].

Definition 1 (Chain). Given some set $T$ and an indexed family $(\equiv_n \subseteq T \times T)_{n \in \mathbb{N}}$ of equivalence relations, a chain is a function $c : \mathbb{N} \to T$ such that $\forall n, m. n \leq m \Rightarrow c(m) \equiv_n c(n)$.

Definition 2. A complete ordered family of equivalences (COFE) is a tuple $(T, (\equiv_n \subseteq T \times T)_{n \in \mathbb{N}}, \lim : \text{chain}(T) \to T)$ satisfying:

- $\forall n. (\equiv_n)$ is an equivalence relation (cofe-equiv)
- $\forall n, m. n \geq m \Rightarrow (\equiv_n) \subseteq (\equiv_m)$ (cofe-mono)
- $\forall x, y. x = y \iff (\forall n. x \equiv_n y)$ (cofe-limit)
- $\forall n, c. \lim(c) \equiv_n c(n)$ (cofe-compl)

The key intuition behind COFEs is that elements $x$ and $y$ are $n$-equivalent, notation $x \equiv_n y$, if they are equivalent for $n$ steps of computation, i.e., if they cannot be distinguished by a program running for no more than $n$ steps. In other words, as $n$ increases, $\equiv_n$ becomes more and more refined (cofe-mono)—and in the limit, it agrees with plain equality (cofe-limit). In order to solve the recursive domain equation in §6 it is also essential that COFEs are complete, i.e., that any chain has a limit (cofe-compl).

Definition 3. An element $x \in T$ of a COFE is called discrete if

$$\forall y \in T. x \equiv_0 y \Rightarrow x = y$$

A COFE $A$ is called discrete if all its elements are discrete. For a set $X$, we write $\Delta X$ for the discrete COFE with $x \equiv_0 x' \equiv x = x'$.

Definition 4. A function $f : T \to U$ between two COFEs is non-expansive (written $f : T \nrightarrow U$) if

$$\forall n, x \in T, y \in T. x \equiv_n y \Rightarrow f(x) \equiv_n f(y)$$

It is contractive if

$$\forall n, x \in T, y \in T. (\forall m < n. x \equiv_m y) \Rightarrow f(x) \equiv_n f(y)$$

Intuitively, applying a non-expansive function to some data will not suddenly introduce differences between seemingly equal data. Elements that cannot be distinguished by programs within $n$ steps remain indistinguishable after applying $f$. The reason that contractive functions are interesting is that for every contractive $f : T \to T$ with $T$ inhabited, there exists a unique fixed-point $\text{fix}(f)$ such that $\text{fix}(f) = f(\text{fix}(f))$.

Definition 5. The category COFE consists of COFEs as objects, and non-expansive functions as arrows.

Note that COFE is cartesian closed. In particular:

Definition 6. Given two COFEs $T$ and $U$, the set of non-expansive functions $\{ f : T \nrightarrow U \}$ is itself a COFE with

$$f \equiv g \iff \forall x \in T. f(x) \equiv_n g(x)$$

Definition 7. A (bi)functor $F : \text{COFE} \to \text{COFE}$ is called locally non-expansive if its action $F_1$ on arrows is itself a non-expansive map. Similarly, $F$ is called locally contractive if $F_1$ is a contractive map.
The function space \((-) \to (-)\) is a locally non-expansive bifunctor. Note that the composition of non-expansive (bi)functors is non-expansive, and the composition of a non-expansive and a contractive (bi)functor is contractive. The reason contractive (bi)functors are interesting is that by America and Rutten’s theorem [1, 3], they have a unique fixed-point.

1.2 RA

**Definition 8.** A resource algebra (RA) is a tuple 
\((M, V \subseteq M, |\cdot| : M \to M^2, (\cdot) : M \times M \to M)\) satisfying:

\[
\forall a, b, c. (a \cdot b) \cdot c = a \cdot (b \cdot c) \quad \text{(RA-ASSOC)}
\]
\[
\forall a, b. a \cdot b = b \cdot a \quad \text{(RA-COMM)}
\]
\[
\forall a. |a| \in M \Rightarrow |a| \cdot a = a \quad \text{(RA-CORE-ID)}
\]
\[
\forall a. |a| \in M \Rightarrow |a| = |a| \quad \text{(RA-CORE-IDEM)}
\]
\[
\forall a, b. |a| \in M \land a \preceq b \Rightarrow |b| \in M \land |a| \preceq |b| \quad \text{(RA-CORE-MONO)}
\]
\[
\forall a, b. (a \cdot b) \in V \Rightarrow a \in V \quad \text{(RA-VALID-OP)}
\]

where \(M^2 \triangleq M \uplus \{\top\}\)

\[
a \preceq b \triangleq \exists c \in M. b = a \cdot c \quad \text{(RA-INCL)}
\]

RAs are closely related to Partial Commutative Monoids (PCMs), with two key differences:

1. The composition operation on RAs is total (as opposed to the partial composition operation of a PCM), but there is a specific subset \(V\) of valid elements that is compatible with the composition operation \((\text{RA-VALID-OP})\).

This take on partiality is necessary when defining the structure of higher-order ghost state, CMRAs, in the next subsection.

2. Instead of a single unit that is an identity to every element, we allow for an arbitrary number of units, via a function \(|\cdot|\) assigning to an element \(a\) its (duplicable) core \(|a|\), as demanded by \((\text{RA-CORE-ID})\). We further demand that \(|\cdot|\) is idempotent \((\text{RA-CORE-IDEM})\) and monotone \((\text{RA-CORE-MONO})\) with respect to the extension order, defined similarly to that for PCMs \((\text{RA-INCL})\).

Notice that the domain of the core is \(M^2\), a set that adds a dummy element \(\top\) to \(M\). Thus, the core can be partial: not all elements need to have a unit. We use the metavariable \(a^2\) to indicate elements of \(M^2\). We also lift the composition \((\cdot)\) to \(M^2\). Partial cores help us to build interesting composite RAs from smaller primitives.

Notice also that the core of an RA is a strict generalization of the unit that any PCM must provide, since \(|\cdot|\) can always be picked as a constant function.

**Definition 9.** It is possible to do a frame-preserving update from \(a \in M\) to \(B \subseteq M\), written \(a \rightsquigarrow B\), if

\[
\forall a^2 \in M^2. a \cdot a^2 \in V \Rightarrow \exists b \in B. b \cdot a^2 \in V
\]

We further define \(a \rightsquigarrow b \triangleq a \rightsquigarrow \{b\}\).

The assertion \(a \rightsquigarrow B\) says that every element \(a^2\) compatible with \(a\) (we also call such elements frames), must also be compatible with some \(b \in B\). Notice that \(a^2\) could be \(\top\), so the frame-preserving update can also be applied to elements that have no frame. Intuitively, this means that whatever assumptions the rest of the program is making about the state of \(\gamma\), if these assumptions are compatible with \(a\), then updating to \(b\) will not invalidate any of these assumptions. Since Iris ensures that the global ghost state is valid, this means that we can soundly update the ghost state from \(a\) to a non-deterministically picked \(b \in B\).

\(^1\)Uniqueness is not proven in Coq.
1.3 CMRA

Definition 10. A CMRA is a tuple \((M : \text{COFE}, (\mathcal{V}_n \subseteq M)_{n \in \mathbb{N}})\), \(\vdash : M \rightarrow M', (\cdot) : M \times M \rightarrow M\) satisfying:

\[
\begin{align*}
\forall n, a, b. & \; a \equiv b \land a \in \mathcal{V}_n \Rightarrow b \in \mathcal{V}_n & \text{(cmra-valid-ne)} \\
\forall n, m. & \; n \geq m \Rightarrow \mathcal{V}_n \subseteq \mathcal{V}_m & \text{(cmra-valid-mono)} \\
\forall a, b, c. & \; (a \cdot b) \cdot c = a \cdot (b \cdot c) & \text{(cmra-assoc)} \\
\forall a, b. & \; a \cdot b = b \cdot a & \text{(cmra-comm)} \\
\forall a. & \; |a| \in M \Rightarrow |a| \cdot a = a & \text{(cmra-core-id)} \\
\forall a. & \; |a| \in M \Rightarrow ||a|| = |a| & \text{(cmra-core-idem)} \\
\forall n, a, b. & \; (a \cdot b) \in \mathcal{V}_n \Rightarrow a \in \mathcal{V}_n & \text{(cmra-valid-op)} \\
\forall n, a, b_1, b_2. & \; a \in \mathcal{V}_n \\
& \quad \Rightarrow \exists c_1, c_2. \; a = c_1 \cdot c_2 \land c_1 \equiv b_1 \land c_2 \equiv b_2 & \text{(cmra-extend)} \\
\end{align*}
\]

where

\[
\begin{align*}
a \preceq b & \triangleq \exists c. \; b = a \cdot c & \text{(cmra-incl)} \\
\equiv n a \preceq b & \triangleq \exists c. \; b \equiv a \cdot c & \text{(cmra-inclN)}
\end{align*}
\]

This is a natural generalization of RAs over COFEs. All operations have to be non-expansive, and the validity predicate \(\mathcal{V}\) can now also depend on the step-index. We define the plain \(\mathcal{V}\) as the “limit” of the \(\mathcal{V}_n\):

\[
\mathcal{V} \triangleq \bigcap_{n \in \mathbb{N}} \mathcal{V}_n
\]

The extension axiom (cmra-extend). Notice that the existential quantification in this axiom is constructive, i.e., it is a sigma type in Coq. The purpose of this axiom is to compute \(a_1, a_2\) completing the following square:

\[
\begin{align*}
\begin{array}{c}
a \\ \parallel \\ a_1 \cdot a_2
\end{array} & \equiv n \begin{array}{c}
b \\ \parallel \\ b_1 \cdot b_2
\end{array}
\end{align*}
\]

where the \(n\)-equivalence at the bottom is meant to apply to the pairs of elements, i.e., we demand \(a_1 \equiv_n b_1\) and \(a_2 \equiv_n b_2\). In other words, extension carries the decomposition of \(b\) into \(b_1\) and \(b_2\) over the \(n\)-equivalence of \(a\) and \(b\), and yields a corresponding decomposition of \(a\) into \(a_1\) and \(a_2\). This operation is needed to prove that \(\triangleright\) commutes with separating conjunction:

\[
\triangleright (P \ast Q) \iff \triangleright P \ast \triangleright Q
\]

Definition 11. An element \(\varepsilon\) of a CMRA \(M\) is called the unit of \(M\) if it satisfies the following conditions:

1. \(\varepsilon\) is valid:
   \(\forall n. \; \varepsilon \in \mathcal{V}_n\)
2. \(\varepsilon\) is a left-identity of the operation:
   \(\forall a \in M. \; \varepsilon \cdot a = a\)
3. \(\varepsilon\) is a discrete COFE element
   \(|\varepsilon| = \varepsilon\)
4. \(\varepsilon\) is its own core:
   \(|\varepsilon| = \varepsilon\)
Lemma 1. If $M$ has a unit $\varepsilon$, then the core $\lvert - \rvert$ is total, i.e., $\forall a. \lvert a \rvert \in M$.

Definition 12. It is possible to do a frame-preserving update from $a \in M$ to $B \subseteq M$, written $a \rightsquigarrow B$, if

$$\forall n, a_i^2. a \cdot a_i^2 \in V_n \Rightarrow \exists b. b \cdot a_i^2 \in V_n$$

We further define $a \rightsquigarrow b \triangleq a \rightsquigarrow \{b\}$.

Note that for RAs, this and the RA-based definition of a frame-preserving update coincide.

Definition 13. A CMRA $M$ is discrete if it satisfies the following conditions:

1. $M$ is a discrete COFE
2. $V$ ignores the step-index:
   $$\forall a \in M. a \in V_0 \Rightarrow \forall n, a \in V_n$$

Note that every RA is a discrete CMRA, by picking the discrete COFE for the equivalence relation. Furthermore, discrete CMRAs can be turned into RAs by ignoring their COFE structure, as well as the step-index of $V$.

Definition 14. A function $f : M_1 \rightarrow M_2$ between two CMRAs is monotone (written $f : M_1 \xrightarrow{\text{mon}} M_2$) if it satisfies the following conditions:

1. $f$ is non-expansive
2. $f$ preserves validity:
   $$\forall n, a \in M_1. a \in V_n \Rightarrow f(a) \in V_n$$
3. $f$ preserves CMRA inclusion:
   $$\forall a \in M_1, b \in M_1. a \preceq b \Rightarrow f(a) \preceq f(b)$$

Definition 15. The category $\mathit{CMRA}$ consists of CMRAs as objects, and monotone functions as arrows.

Note that every object/arrow in $\mathit{CMRA}$ is also an object/arrow of $\mathit{COFE}$. The notion of a locally non-expansive (or contractive) bifunctor naturally generalizes to bifunctors between these categories.
2 COFE constructions

2.1 Next (type-level later)

Given a COFE $\mathcal{T}$, we define $\triangleright T$ as follows (using a datatype-like notation to define the type):

$\triangleright T \triangleq \text{next}(x : T)$

\[
\text{next}(x) \equiv \text{next}(y) \triangleq n = 0 \lor x \overset{n-1}{=} y
\]

Note that in the definition of the carrier $\triangleright T$, \text{next} is a constructor (like the constructors in Coq), i.e., this is short for $\{\text{next}(x) \mid x \in T\}$.

$\triangleright (\cdot)$ is a locally contractive functor from $\text{COFE}$ to $\text{COFE}$.

2.2 Uniform Predicates

Given a CMRA $\mathcal{M}$, we define the COFE $\text{UPred}(\mathcal{M})$ of uniform predicates over $\mathcal{M}$ as follows:

$\text{UPred}(\mathcal{M}) \triangleq \{\varphi : \mathbb{N} \times \mathcal{M} \rightarrow \text{Prop} \mid \forall n,x,y. \varphi(n,x) \land x \overset{n}{=} y \Rightarrow \varphi(n,y) \land (\forall n,m,x,y. \varphi(n,x) \land x \preceq y \land m \leq n \land y \in \mathcal{V}_m \Rightarrow \varphi(m,y))\}$

where $\text{Prop}$ is the set of meta-level propositions, e.g., Coq’s $\text{Prop}$. $\text{UPred}(\cdot)$ is a locally non-expansive functor from $\text{CMRA}$ to $\text{COFE}$.

One way to understand this definition is to re-write it a little. We start by defining the COFE of step-indexed propositions: For every step-index, the proposition either holds or does not hold.

$\text{SProp} \triangleq \wp(\downarrow \mathbb{N})$

$\triangleq \{X \in \wp(\mathbb{N}) \mid \forall n,m.n \geq m \Rightarrow n \in X \Rightarrow m \in X\}$

$X \overset{n}{=} Y \triangleq \forall m \leq n. m \in X \iff m \in Y$

Notice that this notion of $\text{SProp}$ is already hidden in the validity predicate $\mathcal{V}_n$ of a CMRA: We could equivalently require every CMRA to define $\mathcal{V}_n(\cdot) : \mathcal{M} \overset{\text{ne}}{\rightarrow} \text{SProp}$, replacing $\text{CMRA-VALID-NE}$ and $\text{CMRA-VALID-MONO}$.

Now we can rewrite $\text{UPred}(\mathcal{M})$ as monotone step-indexed predicates over $\mathcal{M}$, where the definition of a “monotone” function here is a little funny.

$\text{UPred}(\mathcal{M}) \cong \mathcal{M} \overset{\text{mon}}{\rightarrow} \text{SProp}$

$\triangleq \{\varphi : \mathcal{M} \overset{\text{ne}}{\rightarrow} \text{SProp} \mid \forall n,m,x,y. n \in \varphi(x) \land x \preceq y \land m \leq n \land y \in \mathcal{V}_m \Rightarrow m \in \varphi(y)\}$

The reason we chose the first definition is that it is easier to work with in Coq.
3 RA and CMRA constructions

3.1 Product

Given a family \((M_i)_{i \in I}\) of CMRAs \((I \text{ finite})\), we construct a CMRA for the product \(\prod_{i \in I} M_i\) by lifting everything pointwise.

Frame-preserving updates on the \(M_i\) lift to the product:

\[
\text{prod-update} \quad a \rightsquigarrow_{M_i} B \\
\frac{f[i \mapsto a] \rightsquigarrow \{f[i \mapsto b] \mid b \in B\}}{}
\]

3.2 Sum

The sum CMRA \(M_1 + \perp M_2\) for any CMRAs \(M_1\) and \(M_2\) is defined as (again, we use a datatype-like notation):

\[
M_1 + \perp M_2 \triangleq \text{inl}(a_1 : M_1) \mid \text{inr}(a_2 : M_2) \mid \perp \\
V_n \triangleq \{\text{inl}(a_1) \mid a_1 \in V'_n\} \cup \{\text{inr}(a_2) \mid a_2 \in V''_n\} \\
\text{inl}(a_1) \cdot \text{inl}(b_1) \triangleq \text{inl}(a_1 \cdot b_1) \\
|\text{inl}(a_1)| \triangleq \begin{cases} 
\top & \text{if } |a_1| = \top \\
\text{inl}(|a_1|) & \text{otherwise}
\end{cases}
\]

The composition and core for \(\text{inr}\) are defined symmetrically. The remaining cases of the composition and core are all \(\perp\). Above, \(V'\) refers to the validity of \(M_1\), and \(V''\) to the validity of \(M_2\).

We obtain the following frame-preserving updates, as well as their symmetric counterparts:

\[
\text{sum-update} \quad a \rightsquigarrow_{M_1} B \\
\frac{\text{inl}(a) \rightsquigarrow \{\text{inl}(b) \mid b \in B\}}{}
\]

\[
\text{sum-swap} \quad \forall a, f, n.a, n.a \notin V'_n, b \in V''_n \\
\frac{a \implies \text{inr}(b)}{a \implies \text{inl}(b)}
\]

Crucially, the second rule allows us to swap the “side” of the sum that the CMRA is on if \(V\) has no possible frame.

3.3 Finite partial function

Given some infinite countable \(K\) and some CMRA \(M\), the set of finite partial functions \(K^{\text{fin}} \rightarrow M\) is equipped with a COFE and CMRA structure by lifting everything pointwise.

We obtain the following frame-preserving updates:

\[
\text{fpfn-alloc-strong} \quad G \text{ infinite} \quad a \in V \\
\frac{\emptyset \rightsquigarrow \{[\gamma \mapsto a] \mid \gamma \in G\}}{}
\]

\[
\text{fpfn-alloc} \quad a \in V \\
\frac{\emptyset \rightsquigarrow \{[\gamma \mapsto a] \mid \gamma \in K\}}{}
\]

\[
\text{fpfn-update} \quad a \rightsquigarrow_{M} B \\
\frac{f[i \mapsto a] \rightsquigarrow \{f[i \mapsto b] \mid b \in B\}}{}
\]

Above, \(V\) refers to the validity of \(M\).

\(K^{\text{fin}} \rightarrow (-)\) is a locally non-expansive functor from \(\mathcal{CMRA}\) to \(\mathcal{CMRA}\).
3.4 Agreement

Given some COFE $T$, we define $\text{Ag}(T)$ as follows:

$$\text{Ag}(T) \triangleq \\{(c, V) \in (\mathbb{N} \to T) \times \text{SProp}\} / \sim$$

where $a \sim b \triangleq a.V = b.V \land \forall n. n \in a.V \Rightarrow a.c(n) \equiv b.c(n)$

$$a \equiv b \triangleq (\forall m \leq n. m \in a.V \Leftrightarrow m \in b.V) \land (\forall m \leq n. m \in a.V \Rightarrow a.c(m) \equiv b.c(m))$$

$$\mathcal{V}_n \triangleq \{ a \in \text{Ag}(T) \mid n \in a.V \land \forall m. m \leq n \Rightarrow a.c(m) \equiv b.c(m) \}$$

$$|a| \triangleq a$$

$$a \cdot b \triangleq \{ a.c \mid n \in a.V \land a \equiv b \}$$

$\text{Ag}(\_)$ is a locally non-expansive functor from $\text{COFE}$ to $\text{CMRA}$.

You can think of the $c$ as a chain of elements of $T$ that has to converge only for $n \in V$ steps. The reason we store a chain, rather than a single element, is that $\text{Ag}(T)$ needs to be a COFE itself, so we need to be able to give a limit for every chain of $\text{Ag}(T)$. However, given such a chain, we cannot constructively define its limit: Clearly, the $V$ of the limit is the limit of the $V$ of the chain. But what to pick for the actual data, for the element of $T$? Only if $V = \mathbb{N}$ we have a chain of $T$ that we can take a limit of; if the $V$ is smaller, the chain “cancels”, i.e., stops converging as we reach indices $n \notin V$. To mitigate this, we apply the usual construction to close a set; we go from elements of $T$ to chains of $T$.

We define an injection $\text{ag}$ into $\text{Ag}(T)$ as follows:

$$\text{ag}(x) \triangleq \{ c \triangleq \lambda x. x, V \triangleq \mathbb{N} \}$$

There are no interesting frame-preserving updates for $\text{Ag}(T)$, but we can show the following:

<table>
<thead>
<tr>
<th>$\text{AG-VAL}$</th>
<th>$\text{AG-DUP}$</th>
<th>$\text{AG-AGREE}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{ag}(x) \in \mathcal{V}_n$</td>
<td>$\text{ag}(x) = \text{ag}(x) \cdot \text{ag}(x)$</td>
<td>$\text{ag}(x) \cdot \text{ag}(y) \in \mathcal{V}_n \Rightarrow x \equiv y$</td>
</tr>
</tbody>
</table>

3.5 Exclusive CMRA

Given a COFE $T$, we define a CMRA $\text{Ex}(T)$ such that at most one $x \in T$ can be owned:

$$\text{Ex}(T) \triangleq \text{ex}(T) + \bot$$

$$\mathcal{V}_n \triangleq \{ a \in \text{Ex}(T) \mid a \neq \bot \}$$

All cases of composition go to $\bot$.

$$|\text{ex}(x)| \triangleq \top$$

$$|\bot| \triangleq \bot$$

Remember that $\top$ is the “dummy” element in $M^T$ indicating (in this case) that $\text{ex}(x)$ has no core.

The step-indexed equivalence is inductively defined as follows:

$$\frac{x \equiv y}{\text{ex}(x) \equiv \text{ex}(y)}$$

$$\frac{\bot}{\text{ex}(x) \equiv \text{ex}(y)}$$

$\text{Ex}(\_)$ is a locally non-expansive functor from $\text{COFE}$ to $\text{CMRA}$.

We obtain the following frame-preserving update:

| $\text{EX-UPDATE}$ | $\text{ex}(x) \rightsquigarrow \text{ex}(y)$ |
3.6 STS with tokens

Given a state-transition system (STS, i.e., a directed graph) \((S, \rightarrow \subseteq S \times S)\), a set of tokens \(T\), and a labeling \(L : S \rightarrow \wp(T)\) of protocol-owned tokens for each state, we construct an RA modeling an authoritative current state and permitting transitions given a bound on the current state and a set of locally-owned tokens.

The construction follows the idea of STSs as described in CaReSL [4]. We first lift the transition relation to \(S \times \wp(T)\) (implementing a law of token conservation) and define a stepping relation for the frame of a given token set:

\[
(s, T) \rightarrow (s', T') \triangleq s \rightarrow s' \land L(s) \cup T = L(s') \cup T'
\]

\[
s \xrightarrow{T} s' \triangleq \exists T_1, T_2. T_1 \# L(s) \cup T \land (s, T) \rightarrow (s', T_2)
\]

We further define closed sets of states (given a particular set of tokens) as well as the closure of a set:

\[
\text{closed}(S, T) \triangleq \forall s \in S. L(s) \# T \land (\forall s', s \xrightarrow{T} s' \Rightarrow s' \in S)
\]

\[
\uparrow(S, T) \triangleq \left\{ s' \in S \mid \exists s \in S. s \xrightarrow{T} s' \right\}
\]

The STS RA is defined as follows

\[
M \triangleq \{ \text{auth}((s, T) \in S \times \wp(T)) \mid L(s) \# T \} +
\]

\[
\{ \text{frag}((S, T) \in \wp(S) \times \wp(T)) \mid \text{closed}(S, T) \land S \neq \emptyset \} + \bot
\]

\[
\text{frag}(S_1, T_1) \cdot \text{frag}(S_2, T_2) \triangleq \text{frag}(S_1 \cup S_2, T_1 \cup T_2)
\]

\[
\text{frag}(S, T) \cdot \text{auth}(s, T') \triangleq \text{auth}(s, T') \cdot \text{frag}(S, T) \triangleq \text{auth}(s, T \cup T')
\]

\[
|\text{frag}(S, T)| \triangleq \text{frag}(\uparrow(S, \emptyset), \emptyset)
\]

\[
|\text{auth}(s, T)| \triangleq \text{frag}(\uparrow(s) \setminus \emptyset, \emptyset)
\]

The remaining cases are all \(\bot\).

We will need the following frame-preserving update:

\[
\frac{(s, T) \rightarrow^* (s', T')}{}\quad \frac{\text{closed}(S_2, T_2) \quad S_1 \subseteq S_2 \quad T_2 \subseteq T_1}{\text{frag}(S_1, T_1) \leadsto \text{frag}(S_2, T_2)}
\]

The core is not a homomorphism. The core of the STS construction is only satisfying the RA axioms because we are not demanding the core to be a homomorphism—all we demand is for the core to be monotone with respect the RA-INCL.

In other words, the following does not hold for the STS core as defined above:

\[
|a| \cdot |b| = |a \cdot b|
\]

To see why, consider the following STS:

Now consider the following two elements of the STS RA:

\[
a \triangleq \text{frag}([s_1, s_2], \{T_1\}) \quad b \triangleq \text{frag}([s_1, s_3], \{T_2\})
\]

We have:

\[
a \cdot b = \text{frag}([s_1, T_1, T_2]) \quad |a| = \text{frag}([s_1, s_2, s_4], \emptyset) \quad |b| = \text{frag}([s_1, s_3, s_4], \emptyset)
\]

\[
|a| \cdot |b| = \text{frag}([s_1, s_4], \emptyset) \neq |a \cdot b| = \text{frag}([s_1], \emptyset)
\]

9
4 Language

A language \( \Lambda \) consists of a set \( \text{Expr} \) of expressions (metavariable \( e \)), a set \( \text{Val} \) of values (metavariable \( v \)), and a set \( \text{State} \) of states (metvariable \( \sigma \)) such that

- There exist functions \( \text{val2expr} : \text{Val} \rightarrow \text{Expr} \) and \( \text{expr2val} : \text{Expr} \rightarrow \text{Val} \) (notice the latter is partial), such that
  \[ \forall e, v. \text{expr2val}(e) = v \Rightarrow \text{val2expr}(v) = e \quad \forall v. \text{expr2val} \left( \text{val2expr}(v) \right) = v \]

- There exists a primitive reduction relation
  \[ (\cdot, \cdot \rightarrow \cdot, \cdot, \cdot, \cdot) \subseteq \text{Expr} \times \text{State} \times \text{Expr} \times \text{State} \times (\text{Expr} \cup \{ \bot \}) \]

We will write \( e_1, \sigma_1 \rightarrow e_2, \sigma_2, e_f \) for \( e_1, \sigma_1 \rightarrow e_2, \sigma_2, \bot \).

A reduction \( e_1, \sigma_1 \rightarrow e_2, \sigma_2, e_f \) indicates that, when \( e_1 \) reduces to \( e_2 \), a new thread \( e_f \) is forked off.

- All values are stuck:
  \[ e, \_ \rightarrow \_ \Rightarrow \text{expr2val}(e) = \bot \]

Definition 16. An expression \( e \) and state \( \sigma \) are reducible (written \( \text{red}(e, \sigma) \)) if
  \[ \exists e_2, \sigma_2, e_f. e, \sigma \rightarrow e_2, \sigma_2, e_f \]

Definition 17. An expression \( e \) is said to be atomic if it reduces in one step to a value:
  \[ \forall \sigma_1, e_2, \sigma_2, e_f. e, \sigma \rightarrow e_2, \sigma_2, e_f \Rightarrow \exists v_2. \text{expr2val}(e_2) = v_2 \]

Definition 18 (Context). A function \( K : \text{Expr} \rightarrow \text{Expr} \) is a context if the following conditions are satisfied:

1. \( K \) does not turn non-values into values:
   \[ \forall e. \text{expr2val}(e) = \bot \Rightarrow \text{expr2val}(K(e)) = \bot \]

2. One can perform reductions below \( K \):
   \[ \forall e_1, \sigma_1, e_2, \sigma_2, e_f. e_1, \sigma_1 \rightarrow e_2, \sigma_2, e_f \Rightarrow K(e_1), \sigma_1 \rightarrow K(e_2), \sigma_2, e_f \]

3. Reductions stay below \( K \) until there is a value in the hole:
   \[ \forall e'_1, \sigma_1, e_2, \sigma_2, e_f. \text{expr2val}(e'_1) = \bot \land K(e'_1), \sigma_1 \rightarrow e_2, \sigma_2, e_f \Rightarrow \exists e'_2. e_2 = K(e'_2) \land e'_1, \sigma_1 \rightarrow e'_2, \sigma_2, e_f \]

4.1 Concurrent language

For any language \( \Lambda \), we define the corresponding thread-pool semantics.

Machine syntax

\[ T \in \text{ThreadPool} \triangleq \bigcup_{n} \text{Expr}^{n} \]

Machine reduction

\[
\frac{e_1, \sigma_1 \rightarrow e_2, \sigma_2, e_f \quad e_f \neq \bot}{T \rightarrow \left[ e_1 \right] + + T'; \sigma_1 \rightarrow T \rightarrow \left[ e_2 \right] + + T' + + \left[ e_f \right]; \sigma_2} \quad \frac{e_1, \sigma_1 \rightarrow e_2, \sigma_2}{T \rightarrow \left[ e_1 \right] + + T'; \sigma_1 \rightarrow T \rightarrow \left[ e_2 \right] + + T'; \sigma_2}
\]
5 Logic

To instantiate Iris, you need to define the following parameters:

- A language $\Lambda$, and
- a locally contractive bifunctor $\Sigma : \text{COFE} \to \text{CMRA}$ defining the ghost state, such that for all COFEs $A$, the CMRA $\Sigma(A)$ has a unit. (By Lemma 1, this means that the core of $\Sigma(A)$ is a total function.)

As usual for higher-order logics, you can furthermore pick a signature $S = (T, F, A)$ to add more types, symbols and axioms to the language. You have to make sure that $T$ includes the base types:

$$T \supseteq \{\text{Val}, \text{Expr}, \text{State}, M, \text{InvName}, \text{InvMask}, \text{Prop}\}$$

Elements of $T$ are ranged over by $T$.

Each function symbol in $F$ has an associated arity comprising a natural number $n$ and an ordered list of $n + 1$ types $\tau$ (the grammar of $\tau$ is defined below, and depends only on $T$). We write $F : \tau_1, \ldots, \tau_n \to \tau_{n+1} \in F$ to express that $F$ is a function symbol with the indicated arity.

Furthermore, $A$ is a set of axioms, that is, terms $t$ of type $\text{Prop}$. Again, the grammar of terms and their typing rules are defined below, and depends only on $T$ and $F$, not on $A$. Elements of $A$ are ranged over by $A$.

5.1 Grammar

Syntax. Iris syntax is built up from a signature $S$ and a countably infinite set $\text{Var}$ of variables (ranged over by metavariables $x, y, z$):

$$\tau ::= T | 1 | \tau \times \tau | \tau \to \tau$$

$$t, P, \varphi ::= x | F(t_1, \ldots, t_n) | () | (t, t) | \pi_i t | \lambda x : \tau. t | t(t) | \varepsilon | ||t| | t \cdot t |$$

$$\begin{align*}
\text{False} & | \text{True} | t =_\tau t | P \Rightarrow P | P \land P | P \lor P | P * P | P \rightarrow P | \\
\mu x : \tau. t | \exists x : \tau. P | \forall x : \tau. P |
\end{align*}$$

$$\begin{align*}
\begin{array}{c}
P \upharpoonright_{\tau}^0 \mid \varphi(t) | \text{Phy}(t) | \Box P | P \triangleright P \mid P \triangleright^t P | \text{wp}_t t \{x. t\}
\end{array}
\end{align*}$$

Recursive predicates must be guarded: in $\mu x. t$, the variable $x$ can only appear under the later $\triangleright$ modality.

Note that $\Box$ and $\triangleright$ bind more tightly than $\ast, \neg \ast, \land, \lor, \Rightarrow$. We will write $P \triangleright_{\tau} t$ for $P \triangleright_{\tau}^t P$. If we omit the mask, then it is $\top$ for weakest precondition $\text{wp} e \{x. P\}$ and $\emptyset$ for primitive view shifts $\triangleright P$.

Some propositions are timeless, which intuitively means that step-indexing does not affect them. This is a meta-level assertion about propositions, defined as follows:

$$\Gamma \vdash \text{timeless}(P) \triangleq \Gamma \triangleright P \vdash P \lor \triangleright \text{False}$$

Metavariable conventions. We introduce additional metavariables ranging over terms and generally let the choice of metavariable indicate the term’s type:

<table>
<thead>
<tr>
<th>metavariable</th>
<th>type</th>
<th>metavariable</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t, u$</td>
<td>arbitrary</td>
<td>$i$</td>
<td>InvName</td>
</tr>
<tr>
<td>$v, w$</td>
<td>Val</td>
<td>$\mathcal{E}$</td>
<td>InvMask</td>
</tr>
<tr>
<td>$e$</td>
<td>Expr</td>
<td>$a, b$</td>
<td>M</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>State</td>
<td>$P, Q, R$</td>
<td>Prop</td>
</tr>
<tr>
<td>$\varphi, \psi, \zeta$</td>
<td>$\tau \to \text{Prop}$ (when $\tau$ is clear from context)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Variable conventions. We assume that, if a term occurs multiple times in a rule, its free variables are exactly those binders which are available at every occurrence.

5.2 Types

Iris terms are simply-typed. The judgment $\Gamma \vdash t : \tau$ expresses that, in variable context $\Gamma$, the term $t$ has type $\tau$.

A variable context, $\Gamma = x_1 : \tau_1, \ldots, x_n : \tau_n$, declares a list of variables and their types. In writing $\Gamma, x : \tau$, we presuppose that $x$ is not already declared in $\Gamma$.

Well-typed terms

<table>
<thead>
<tr>
<th>[ x : \tau \vdash x : \tau ]</th>
<th>[ \Gamma \vdash t : \tau ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \Gamma, x : \tau' \vdash t : \tau ]</td>
<td>[ \Gamma, x : \tau', y : \tau' \vdash t : \tau ]</td>
</tr>
<tr>
<td>[ \Gamma, x : \tau' \vdash t[x/y] : \tau ]</td>
<td>[ \Gamma, x : \tau', y : \tau'' \vdash \tau_1, \ldots, \tau_n \rightarrow \tau_{n+1} \in \mathcal{F} ]</td>
</tr>
<tr>
<td>[ \Gamma \vdash F(t_1, \ldots, t_n) : \tau_{n+1} ]</td>
<td>[ \Gamma \vdash () : 1 ]</td>
</tr>
<tr>
<td>[ \Gamma \vdash t : \tau_1 ]</td>
<td>[ \Gamma \vdash t : \tau_1 \times \tau_2 ]</td>
</tr>
<tr>
<td>[ \Gamma \vdash \tau_i t_i : \tau_i ]</td>
<td>[ \Gamma \vdash \pi_1 t : \tau ]</td>
</tr>
<tr>
<td>[ \Gamma \vdash \lambda x. t : \tau \rightarrow \tau' ]</td>
<td>[ \Gamma \vdash t : \tau \rightarrow \tau' ]</td>
</tr>
<tr>
<td>[ \Gamma \vdash t(u) : \tau' ]</td>
<td>[ \Gamma \vdash \varepsilon : M ]</td>
</tr>
<tr>
<td>[ \Gamma \vdash a : M ]</td>
<td>[ \Gamma \vdash a \cdot b : M ]</td>
</tr>
</tbody>
</table>

\[ \Gamma \vdash \text{False} : \text{Prop} \] \[ \Gamma \vdash \text{True} : \text{Prop} \]

| \[ \Gamma \vdash P : \text{Prop} \] | \[ \Gamma \vdash Q : \text{Prop} \] |
| \[ \Gamma \vdash \top : \text{Prop} \] | \[ \Gamma \vdash \top : \text{Prop} \] |
| \[ \Gamma \vdash P \land Q : \text{Prop} \] | \[ \Gamma \vdash P \lor Q : \text{Prop} \] |
| \[ \Gamma \vdash (\top \lor Q) : \text{Prop} \] | \[ \Gamma \vdash P \rightarrow Q : \text{Prop} \] |

| \[ \Gamma, x : \tau \vdash t : \tau \] | \[ \Gamma, x : \tau \vdash \mu x. t : \tau \] |
| \[ \Gamma \vdash t : \tau \rightarrow \tau \] | \[ \Gamma \vdash \exists x. \tau. P : \text{Prop} \] |
| \[ \Gamma \vdash \forall x. \tau. P : \text{Prop} \] | \[ \Gamma \vdash \text{InvName} \] |
| \[ \Gamma \vdash a : \tau \] | \[ \Gamma \vdash a : M \] |

| \[ \Gamma \vdash \mathcal{V}(a) : \text{Prop} \] | \[ \Gamma \vdash \text{State} \] |
| \[ \Gamma \vdash P : \text{Prop} \] | \[ \Gamma \vdash \Box P : \text{Prop} \] |
| \[ \Gamma \vdash P : \text{Prop} \] | \[ \Gamma \vdash \\ \rightarrow P : \text{Prop} \] |

| \[ \Gamma \vdash a : \tau \] | \[ \tau \text{ is a CMRA} \] |

| \[ \Gamma \vdash \sigma : \text{State} \] | \[ \Gamma \vdash P : \text{Prop} \] |
| \[ \Gamma \vdash \Phi(\sigma) : \text{Prop} \] | \[ \Gamma \vdash P : \text{Prop} \] |
| \[ \Gamma \vdash \text{InvMask} \] | \[ \Gamma \vdash \text{InvMask} \] |
| \[ \Gamma \vdash \mathcal{E} \rightarrow \mathcal{E}' P : \text{Prop} \] | \[ \Gamma \vdash \mathcal{E} \rightarrow \mathcal{E}' P : \text{Prop} \] |

5.3 Proof rules

The judgment $\Gamma \vdash \Theta \vdash P$ says that with free variables $\Gamma$, proposition $P$ holds whenever all assumptions $\Theta$ hold. We implicitly assume that an arbitrary variable context, $\Gamma$, is added to every constituent of the rules. Furthermore, an arbitrary boxed assertion context $\Box \Theta$ may be added to every constituent. Axioms $\Gamma \vdash P \vdash Q$ indicate that both $\Gamma \vdash P$ and $\Gamma \vdash Q \vdash P$ can be derived.
Laws of intuitionistic higher-order logic with equality. This is entirely standard.

\[\begin{array}{c|c|c|c|c|c|c}
\text{ASM} & \text{EQ} & \text{REFL} & \text{LE} & \top \text{I} & \land \text{I} \\
P \in \Theta & \Theta \vdash P & \Theta \vdash t = t' & \Theta \vdash \top = \top & \Theta \vdash \top & \Theta \vdash P \wedge Q \\
\hline
\Theta \vdash P & \Theta \vdash P'[t'/t] & \Theta \vdash t = t & \Theta \vdash P & \Theta \vdash P & \Theta \vdash P \land Q \\
\hline
\land \text{EL} & \land \text{ER} & \lor \text{IL} & \lor \text{IR} & \top \text{E} & \land \text{ER} \\
\Theta \vdash P \land Q & \Theta \vdash P \land Q & \Theta \vdash P \lor Q & \Theta \vdash P \lor Q & \Theta \vdash P \lor Q & \Theta \vdash P \land Q \\
\hline
\Rightarrow \text{I} & \Rightarrow \text{E} & \forall \text{I} & \forall \text{E} & \exists \text{I} & \exists \text{E} \\
\Rightarrow \Theta, P \vdash Q & \Rightarrow \Theta \vdash P \Rightarrow Q & \Theta \vdash P & \Theta \vdash P & \forall \Gamma, x : \tau | \Theta \vdash \top & \forall \Gamma, x : \tau, P \vdash t : \tau \\
\hline
\exists \text{I} & \exists \text{E} \\
\Gamma | \Theta \vdash P'[t/x] & \Gamma | \Theta \vdash \exists x : \tau, P \vdash t : \tau \\
\hline
\end{array}\]

Furthermore, we have the usual \(\eta\) and \(\beta\) laws for projections, \(\lambda\) and \(\mu\).

Laws of (affine) bunched implications.

\[\begin{array}{c}
\text{True} \ast P \vdash P \\
P \ast Q \vdash Q \ast P \\
(P \ast Q) \ast R \vdash (P \ast (Q \ast R)) \\
\hline
\ast \text{-mono} \\
P_1 \vdash Q_1 & P_2 \vdash Q_2 \\
P_1 \ast P_2 \vdash Q_1 \ast Q_2 \\
P \vdash Q \vdash R \\
\hline
\Rightarrow \text{-I-E} \\
P \ast Q \vdash R \\
\end{array}\]

Laws for ghosts and physical resources.

\[\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array{a}
\( t \) or \( t' \) is a discrete COFE element
\[ \text{timeless}(t = t') \]
\[ a \text{ is an element of a discrete CMRA} \]
\[ \text{timeless}(\forall(a)) \]
\[ \text{a is an element of a discrete CMRA} \]
\[ \text{timeless}(\forall(a)) \]
\[ \Gamma \vdash \text{timeless}(Q) \]
\[ \Gamma \vdash \text{timeless}(P \Rightarrow Q) \]
\[ \Gamma, x : \tau \vdash \text{timeless}(P) \]
\[ \Gamma, x : \tau \vdash \text{timeless}(\forall x : \tau. P) \]
\[ \Gamma \vdash \text{timeless}(\exists x : \tau. P) \]

Laws for the always modality.

\[ \Box \]
\[ \Box \Theta \vdash P \]
\[ \Box E \]
\[ \Box P \vdash P \]
\[ \Box P \land Q \vdash \Box (P \land Q) \]
\[ \Box P \lor Q \vdash \Box (P \lor Q) \]
\[ \Box \top \vdash \Box \top \]
\[ \Box \bot \vdash \Box \bot \]
\[ t =_\tau t' \vdash \Box t =_\tau t' \]
\[ P \vdash \Box P \]
\[ \Box[a/x] \vdash \Box[a/x] P \]
\[ \forall(a) \vdash \Box \forall(a) \]

Laws of primitive view shifts.

\[ \text{PVS-INTRO} \]
\[ P \vdash \Box P \]
\[ \text{PVS-MONO} \]
\[ P \vdash Q \]
\[ \Box P \vdash \Box Q \]
\[ \Box (P \land Q) \vdash \Box (P \land Q) \]
\[ \Box (P \lor Q) \vdash \Box (P \lor Q) \]
\[ \Box \top \vdash \Box \top \]
\[ \Box \bot \vdash \Box \bot \]
\[ \Box \Theta \vdash \Box P \]
\[ \text{PVS-MASK-FRAME} \]
\[ \Box E \]
\[ \Box \Box P \vdash \Box P \]
\[ \Box (P \land Q) \vdash \Box (P \land Q) \]
\[ \Box (P \lor Q) \vdash \Box (P \lor Q) \]
\[ \Box \top \vdash \Box \top \]
\[ \Box \bot \vdash \Box \bot \]
\[ \Box \Theta \vdash \Box P \]
\[ \text{PVS-OPENI} \]
\[ P \vdash \Box P \]
\[ \Box P \land \Box P \vdash \Box P \]
\[ \Box P \lor \Box P \vdash \Box P \]
\[ \Box \top \vdash \Box \top \]
\[ \Box \bot \vdash \Box \bot \]
\[ \Box \Theta \vdash \Box P \]
\[ \text{PVS-CLOSEI} \]
\[ P \vdash \Box P \]
\[ \Box P \land \Box P \vdash \Box P \]
\[ \Box P \lor \Box P \vdash \Box P \]
\[ \Box \top \vdash \Box \top \]
\[ \Box \bot \vdash \Box \bot \]
\[ \Box \Theta \vdash \Box P \]
\[ \text{PVS-UPDATE} \]
\[ a \rightsquigarrow B \]
\[ \Box[a/x] \vdash \Box \exists b \in B \]

Laws of weakest preconditions.

\[ \text{WP-VALUE} \]
\[ P[v/x] \vdash \text{wp}_E v \{x. P\} \]
\[ \text{WP-MONO} \]
\[ E_1 \subseteq E_2 \]
\[ x : \text{val} \mid P \vdash Q \]
\[ \text{wp}_{E_1} e \{x. P\} \vdash \text{wp}_{E_2} e \{x. Q\} \]
\[ \text{WP-ATOMIC} \]
\[ \text{atomic}(e) \]
\[ E_1 \vdash \text{wp}_{E_2} e \{x. E_2 \Box P\} \]
\[ E_2 \subseteq E_1 \]
\[ \text{wp}_{E_2} e \{x. E_2 \Box P\} \vdash \text{wp}_{E_1} e \{x. P\} \]
\[ \text{WP-FRAME} \]
\[ Q \land \text{wp}_{E_2} e \{x. P\} \vdash \text{wp}_{E_2} e \{x. Q \land P\} \]
\[ \text{WP-ATOM} \]
\[ \text{atomic}(e) \]
\[ E_1 \vdash \text{wp}_{E_2} e \{x. E_2 \Box P\} \]
\[ E_2 \subseteq E_1 \]
\[ \text{wp}_{E_2} e \{x. E_2 \Box P\} \vdash \text{wp}_{E_1} e \{x. P\} \]
\[ \text{WP-FRAME-STEP} \]
\[ \text{expr2val}(e) = \bot \]
\[ Q \land \text{wp}_{E_2} e \{x. E_2 \Box P\} \vdash \text{wp}_{E_2} e \{x. Q \land P\} \]
\[ \text{WP-BIND} \]
\[ K \text{ is a context} \]
\[ \text{wp}_{E} e \{x. \text{wp}_{E} K(\text{val2expr}(x)) \{y. P\}\} \vdash \text{wp}_{E} K(e) \{y. P\} \]
Lifting of operational semantics.

\[ \text{wp-lift-step} \]
\[ \mathcal{E}_2 \subseteq \mathcal{E}_1 \quad \text{expr2val}(e_1) = \bot \]
\[ \vdash \text{wp}_{\mathcal{E}_1} e_1 \{ x. P \} \]

Notice that primitive view shifts cover everything to their right, i.e., \( \Longrightarrow P \ast Q \equiv \Longrightarrow (P \ast Q) \).

Here we define \( \text{wp}_{\mathcal{E}} e_1 \{ x. P \} \equiv \text{True} \) if \( e_1 = \bot \) (remember that our stepping relation can, but does not have to, define a forked-off expression).

### 5.4 Adequacy

The adequacy statement concerning functional correctness reads as follows:

\[ \forall \mathcal{E}, e, v, \varphi, \sigma, a, \sigma', T'. \]
\[ (\forall n. a \in \mathcal{V}_n) \Rightarrow \]
\[ (\text{Phy}(\sigma) \ast \mathcal{V}_n \vdash \text{wp}_{\mathcal{E}} e \{ x. \varphi(x) \}) \Rightarrow \]
\[ \sigma; [e] \rightarrow^* \sigma'; [v] + T' \Rightarrow \]
\[ \varphi(v) \]

where \( \varphi \) is a meta-level predicate over values, i.e., it can mention neither resources nor invariants.

Furthermore, the following adequacy statement shows that our weakest preconditions imply that the execution never gets stuck: Every expression in the thread pool either is a value, or can reduce further.

\[ \forall \mathcal{E}, e, \sigma, a, \sigma', T'. \]
\[ (\forall n. a \in \mathcal{V}_n) \Rightarrow \]
\[ (\text{Phy}(\sigma) \ast \mathcal{V}_n \vdash \text{wp}_{\mathcal{E}} e \{ x. \varphi(x) \}) \Rightarrow \]
\[ \sigma; [e] \rightarrow^* \sigma'; T' \Rightarrow \]
\[ \forall e' \in T'. \text{expr2val}(e') \neq \bot \lor \text{red}(e', \sigma') \]

Notice that this is stronger than saying that the thread pool can reduce; we actually assert that every non-finished thread can take a step.
6 Model and semantics

The semantics closely follows the ideas laid out in [2].

6.1 Generic model of base logic

The base logic including equality, later, always, and a notion of ownership is defined on \( \text{UPred}(M) \) for any CMRA \( M \).

Interpretation of base assertions

\[
\begin{align*}
\llbracket \Gamma \vdash t =, u : \text{Prop} \rrbracket_\gamma &\triangleq \lambda_. \quad \{ n \mid \llbracket \Gamma \vdash t : \tau \rrbracket_\gamma \triangleq \llbracket \Gamma \vdash u : \tau \rrbracket_\gamma \} \\
\llbracket \Gamma \vdash \text{False} : \text{Prop} \rrbracket_\gamma &\triangleq \lambda_. \\emptyset \\
\llbracket \Gamma \vdash \text{True} : \text{Prop} \rrbracket_\gamma &\triangleq \lambda_. \\mathbb{N} \\
\llbracket \Gamma \vdash P \land Q : \text{Prop} \rrbracket_\gamma &\triangleq \lambda a. \llbracket \Gamma \vdash P : \text{Prop} \rrbracket_\gamma(a) \cap \llbracket \Gamma \vdash Q : \text{Prop} \rrbracket_\gamma(a) \\
\llbracket \Gamma \vdash P \lor Q : \text{Prop} \rrbracket_\gamma &\triangleq \lambda a. \llbracket \Gamma \vdash P : \text{Prop} \rrbracket_\gamma(a) \cup \llbracket \Gamma \vdash Q : \text{Prop} \rrbracket_\gamma(a) \\
\llbracket \Gamma \vdash P \Rightarrow Q : \text{Prop} \rrbracket_\gamma &\triangleq \lambda a. \quad \{ n \mid \forall m, b. m \leq n \land a \leq b \land b \in \mathcal{V}_m \Rightarrow \\
&\quad \quad \quad m \in \llbracket \Gamma \vdash P : \text{Prop} \rrbracket_\gamma(b) \Rightarrow \\
&\quad \quad \quad m \in \llbracket \Gamma \vdash Q : \text{Prop} \rrbracket_\gamma(b) \} \\
\llbracket \Gamma \vdash \forall x : \tau. P : \text{Prop} \rrbracket_\gamma &\triangleq \lambda a. \quad \{ n \mid \forall v \in \llbracket \tau \rrbracket. n \in \llbracket \Gamma, x : \tau \vdash P : \text{Prop} \rrbracket_\gamma[x:=v](a) \} \\
\llbracket \Gamma \vdash \exists x : \tau. P : \text{Prop} \rrbracket_\gamma &\triangleq \lambda a. \quad \{ n \mid \exists v \in \llbracket \tau \rrbracket. n \in \llbracket \Gamma, x : \tau \vdash P : \text{Prop} \rrbracket_\gamma[x:=v](a) \} \\
\llbracket \Gamma \vdash \Box P : \text{Prop} \rrbracket_\gamma &\triangleq \lambda a. \llbracket \Gamma \vdash P : \text{Prop} \rrbracket_\gamma(|a|) \\
\llbracket \Gamma \vdash \triangleright P : \text{Prop} \rrbracket_\gamma &\triangleq \lambda a. \quad \{ n \mid n = 0 \lor n - 1 \in \llbracket \Gamma \vdash P : \text{Prop} \rrbracket_\gamma(a) \} \\
\llbracket \Gamma \vdash P * Q : \text{Prop} \rrbracket_\gamma &\triangleq \lambda a. \quad \{ n \mid \exists b_1, b_2. a \triangleq b_1 \land b_2 \land \\
&\quad \quad \quad n \in \llbracket \Gamma \vdash P : \text{Prop} \rrbracket_\gamma(b_1) \land n \in \llbracket \Gamma \vdash Q : \text{Prop} \rrbracket_\gamma(b_2) \} \\
\llbracket \Gamma \vdash P \rightarrow Q : \text{Prop} \rrbracket_\gamma &\triangleq \lambda a. \quad \{ n \mid \forall m, b. m \leq n \land a \cdot b \in \mathcal{V}_m \Rightarrow \\
&\quad \quad \quad m \in \llbracket \Gamma \vdash P : \text{Prop} \rrbracket_\gamma(b) \Rightarrow \\
&\quad \quad \quad m \in \llbracket \Gamma \vdash Q : \text{Prop} \rrbracket_\gamma(a \cdot b) \} \\
\llbracket \Gamma \vdash \text{Own}(a) : \text{Prop} \rrbracket_\gamma &\triangleq \lambda b. \quad \{ n \mid \llbracket \Gamma \vdash a : M \rrbracket \triangleq b \} \\
\llbracket \Gamma \vdash \mathcal{V}(a) : \text{Prop} \rrbracket_\gamma &\triangleq \lambda_. \quad \{ n \mid \llbracket \Gamma \vdash a : \tau \rrbracket \in \mathcal{V}_n \}
\end{align*}
\]

For every definition, we have to show all the side-conditions: The maps have to be non-expansive and monotone.

6.2 Iris model

Semantic domain of assertions. The first complicated task in building a model of full Iris is defining the semantic model of \( \text{Prop} \). We start by defining the functor that assembles the CMRAs
we need to the global resource CMRA:

$$ResF(T^{op}, T) \triangleq \left\{ w : \mathbb{N} \xrightarrow{\text{fin}} \text{AG}(\triangleright T), \pi : \text{EX}(\text{State})^+, g : \Sigma(T^{op}, T) \right\}$$

Above, $M^\square$ is the monoid obtained by adding a unit to $M$. (It’s not a coincidence that we used the same notation for the range of the core; it’s the same type either way: $M + 1$.) Remember that $\Sigma$ is the user-chosen bifunctor from $\text{COFE}$ to $\text{CMRA}$ (see §5). $ResF(T^{op}, T)$ is a CMRA by lifting the individual CMRAs pointwise. Furthermore, since $\Sigma$ is locally contractive, so is $ResF$.

Now we can write down the recursive domain equation:

$$\text{iPreProp} \equiv \text{UPred}(\text{ResF}(\text{iPreProp}, \text{iPreProp}))$$

$i\text{PreProp}$ is a COFE defined as the fixed-point of a locally contractive bifunctor. This fixed-point exists and is unique by America and Rutten’s theorem [1, 3]. We do not need to consider how the object is constructed. We only need the isomorphism, given by

$$\text{Res} \triangleq \text{ResF}(\text{iPreProp}, \text{iPreProp})$$

$$\text{iProp} \triangleq \text{UPred}(\text{Res})$$

$$\xi : \text{iProp} \xrightarrow{\text{nc}} \text{iPreProp}$$

$$\xi^{-1} : \text{iPreProp} \xrightarrow{\text{nc}} \text{iProp}$$

We then pick $\text{iProp}$ as the interpretation of $\text{Prop}$:

$$\llbracket \text{Prop} \rrbracket \triangleq \text{iProp}$$

**Interpretation of assertions.** $\text{iProp}$ is a $\text{UPred}$, and hence the definitions from §6.1 apply. We only have to define the interpretation of the missing connectives, the most interesting bits being primitive view shifts and weakest preconditions.

**World satisfaction**

$$\neg \models \_ \models \_ : \Delta \text{State} \times \Delta \wp(\mathbb{N}) \times \text{Res} \xrightarrow{\text{nc}} \text{SProp}$$

$$\text{pre-wsat}(n, E, \sigma, R, r) \triangleq r \in \mathcal{V}(n+1) \land r \pi = \text{ex}(\sigma) \land \text{dom}(R) \subseteq E \cap \text{dom}(r.w) \land$$

$$\forall i \in E, P \in \text{iProp}. (r.w)(i) \xrightarrow{\text{nc}} \text{ag}(\xi(P)) \Rightarrow n \in \text{P}(R(i))$$

$$\sigma \models_E r \triangleq \{0\} \cup \left\{ n + 1 \mid \exists R : \mathbb{N} \xrightarrow{\text{fin}} \text{Res}. \text{pre-wsat}(n, E, \sigma, R, r \cdot \bigcap_i R(i)) \right\}$$

**Primitive view-shift**

$$\text{pus}_{\xi}(\_)(P) = \lambda r. \left\{ r \cap k, E, \sigma. 0 \leq k \leq n \land (E_1 \cup E_2) \# E \land k \in \sigma \models_{E \cup E_2} r \cdot r_l \Rightarrow \exists s. k \in P(s) \land k \in \sigma \models_{E_2 \cup E_2} s \cdot r_l \right\}$$

**Weakest precondition**

$$\text{wp}_{\_}(\_)(E, e, \varphi) \triangleq \lambda r. \left\{ \forall r_l, m, E, \sigma. 0 \leq m < n \land \sigma \# E \land m + 1 \in \sigma \models_{E \cup E_2} r \cdot r_l \Rightarrow \right\}$$

$$\left\{ \forall e, \text{expr2val}(e) = v \Rightarrow \exists s. m + 1 \in \varphi(v)(s) \land m + 1 \in \sigma \models_{E \cup E_2} s \cdot r_l \land (\forall v. \text{expr2val}(e) = v \Rightarrow \exists s. m + 1 \in \varphi(v)(s) \land m + 1 \in \sigma \models_{E \cup E_2} s \cdot r_l) \land \right\}$$

$$\left\{ \exists s_1, s_2. m \in \sigma \models_{E \cup E_2} s_1 \cdot s_2 \cdot r_l \land m \in \text{wp}(E, e_2, \varphi)(s_1) \land \right\}$$

$$\left( e_1 = \perp \land m \in \text{wp}(T, e_1, \lambda_r. \lambda_r. \mathbb{N})(s_2) \right)$$

$$\text{wp}(E, e, \varphi) \triangleq \text{fix}(\text{pre-wp})(E, e, \varphi)$$
For the remaining base types define
\[
\nu, \varepsilon, a \ni \text{Own}(\xi^P, \varepsilon, \varepsilon)
\]
\[
\text{Phy}(\sigma) \ni \text{Own}(\varepsilon, \varepsilon, a)
\]

Interpretation of non-propositional terms

\[
\Gamma \vdash t : \text{Prop} \ni \text{wp} \ni \text{InvName} \ni \text{InvMask}
\]
\[
\Gamma \vdash \text{wp}_e e \ni \text{Prop} \ni \text{wp} \ni \text{Expr} \ni \text{Val}
\]
\[
\Gamma \vdash P \ni \text{Prop} \ni \text{wp} \ni \text{Expr} \ni \text{Val}
\]

Remaining semantic domains, and interpretation of non-assertion terms.

The remaining domains are interpreted as follows:

\[
\|\nu\| \ni \Delta\nu
\]
\[
\|\varepsilon\| \ni \Delta\varepsilon
\]
\[
\|\text{Val}\| \ni \Delta\text{Val}
\]
\[
\|\text{Expr}\| \ni \Delta\text{Expr}
\]
\[
\|\text{State}\| \ni \Delta\text{State}
\]
\[
\|\tau \\rightarrow \tau'\| \ni \tau \ni \text{Prop} \ni \text{Val}
\]

For the remaining base types \(\tau\) defined by the signature \(\mathcal{S}\), we pick an object \(X_\tau\) in \(\text{COFE}\) and define

\[
\|\tau\| \ni X_\tau
\]

For each function symbol \(F : \tau_1, \ldots, \tau_n \rightarrow \tau_{n+1} \ni \mathcal{F}\), we pick a function \(\|F\| : \|\tau_1\| \ni \cdots \ni \|\tau_n\| \ni \|\tau_{n+1}\|

Interpretation of non-propositional terms

\[
\Gamma \vdash t : \tau \ni \text{wp} \ni \text{Expr} \ni \text{Val}
\]

An environment \(\Gamma\) is interpreted as the set of finite partial functions \(\rho\), with \(\text{dom}(\rho) = \text{dom}(\Gamma)\) and \(\rho(x) \in \|\Gamma(x)\|

Logical entailment. We can now define semantic logical entailment.
Interpretation of entailment

\[ [[\Gamma \mid \Theta \vdash P]] : \text{Prop} \]

\[ [[\Gamma \mid \Theta \vdash P]] \triangleq \forall n \in \mathbb{N} \cdot \forall r \in \text{Res} \cdot \forall \gamma \in [[\Gamma]], \]
\[ (\forall Q \in \Theta. \ n \in [[\Gamma \vdash Q : \text{Prop}]_\gamma(r)) \rightarrow n \in [[\Gamma \vdash P : \text{Prop}]_\gamma(r) \]

The soundness statement of the logic reads

\[ \Gamma \mid \Theta \vdash P \Rightarrow [[\Gamma \mid \Theta \vdash P]] \]
7 Derived proof rules and other constructions

We will below abuse notation, using the term meta-variables like \( v \) to range over (bound) variables of the corresponding type. We omit type annotations in binders and equality, when the type is clear from context. We assume that the signature \( S \) embeds all the meta-level concepts we use, and their properties, into the logic. (The Coq formalization is a shallow embedding of the logic, so we have direct access to all meta-level notions within the logic anyways.)

7.1 Base logic

We collect here some important and frequently used derived proof rules.

\[
\begin{align*}
P \Rightarrow Q & \vdash P \quad \vdash  \\
P * \exists x. Q & \vdash \exists x. P * Q \\
P * \forall x. Q & \vdash \forall x. P * Q \\
\Box (P * Q) & \vdash \Box P * \Box Q
\end{align*}
\]

\[
\begin{align*}
\vdash (P \Rightarrow Q) & \vdash \Box P \Rightarrow \Box Q \\
\vdash (P \Rightarrow Q) & \vdash \Box P \Rightarrow \Box Q \\
\vdash (P \Rightarrow Q) & \vdash \Box (P \Rightarrow Q)
\end{align*}
\]

\[
\begin{align*}
\Theta \vdash P & \vdash \Box P \\
\Theta \vdash P & \vdash \Box P
\end{align*}
\]

Persistent assertions.

Definition 19. An assertion \( P \) is persistent if \( P \vdash \Box P \).

Of course, \( \Box P \) is persistent for any \( P \). Furthermore, by the proof rules given in §5.3, \( t = t' \) as well as \( \forall (a) \) and \( [P] \) are persistent. Persistence is preserved by conjunction, disjunction, separating conjunction as well as universal and existential quantification.

In our proofs, we will implicitly add and remove \( \Box \) from persistent assertions as necessary, and generally treat them like normal, non-linear assumptions.

Timeless assertions. We can show that the following additional closure properties hold for timeless assertions:

\[
\begin{align*}
\Gamma \vdash \text{timeless}(P) & \quad \Gamma \vdash \text{timeless}(Q) \\
\Gamma \vdash \text{timeless}(P \land Q) & \\
\Gamma \vdash \text{timeless}(P) & \quad \Gamma \vdash \text{timeless}(Q) \\
\Gamma \vdash \text{timeless}(P * Q) & \quad \Gamma \vdash \text{timeless}(\Box P)
\end{align*}
\]

7.2 Program logic

Hoare triples and view shifts are syntactic sugar for weakest (liberal) preconditions and primitive view shifts, respectively:

\[
\begin{align*}
\{P\} e \{v. Q\}_{E} & \triangleq \Box (P \Rightarrow \text{wp}_{E} e \{\lambda v. Q\}) \\
& \quad \quad P \ [E_{1} \Rightarrow E_{2}] Q \ [E_{1} \Rightarrow E_{2}] \ (P \Rightarrow \Box E_{1} \Rightarrow E_{2} Q) \\
& \quad \quad P \ [E_{1} \Rightarrow E_{2}] Q \ [E_{1} \Rightarrow E_{2}] \ (P \Rightarrow \Box E_{1} \Rightarrow E_{2} Q)
\end{align*}
\]

We write just one mask for a view shift when \( E_{1} = E_{2} \). Clearly, all of these assertions are persistent. The convention for omitted masks is similar to the base logic: An omitted \( E \) is \( \top \) for Hoare triples and \( \emptyset \) for view shifts.
**View shifts.** The following rules can be derived for view shifts.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VS-UPDATE</td>
<td>(a \rightarrow B)</td>
</tr>
<tr>
<td>VS-MASK-FRAME</td>
<td>(P \xi \Rightarrow \xi' Q)</td>
</tr>
<tr>
<td>VS-OPENI</td>
<td>([P] \vdash \text{True}) (\xi \Rightarrow \emptyset P)</td>
</tr>
<tr>
<td>VS-EXIST</td>
<td>(\forall x. (P \xi \Rightarrow \xi' Q))</td>
</tr>
<tr>
<td>VS-OPENI</td>
<td>([P] \vdash \theta \Rightarrow \emptyset P)</td>
</tr>
<tr>
<td>VS-CLOSEI</td>
<td>(\emptyset \Rightarrow \theta) (\xi \Rightarrow \emptyset P)</td>
</tr>
<tr>
<td>VS-BOX</td>
<td>(\square Q \vdash P \xi \Rightarrow \xi' R)</td>
</tr>
</tbody>
</table>
| VS-ALLOCI | \(\xi \Rightarrow \xi' P\)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VS-TRANS</td>
<td>(P \xi \Rightarrow \xi' Q) (\xi' \Rightarrow \xi' R) (\xi \subseteq \xi' \cup \xi')</td>
</tr>
<tr>
<td>VS-FRAME</td>
<td>(P \xi \Rightarrow \xi' Q \Rightarrow \xi' R)</td>
</tr>
<tr>
<td>VS-DISJ</td>
<td>(P \xi \Rightarrow \xi' Q \Rightarrow \xi' R)</td>
</tr>
</tbody>
</table>
| VS-ALLOCI | \(\xi \Rightarrow \xi' P\)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VS-UPDATE</td>
<td>(a \rightarrow B)</td>
</tr>
<tr>
<td>VS-MASK-FRAME</td>
<td>(P \xi \Rightarrow \xi' Q)</td>
</tr>
<tr>
<td>VS-OPENI</td>
<td>([P] \vdash \text{True}) (\xi \Rightarrow \emptyset P)</td>
</tr>
<tr>
<td>VS-CLOSEI</td>
<td>(\emptyset \Rightarrow \theta) (\xi \Rightarrow \emptyset P)</td>
</tr>
<tr>
<td>VS-BOX</td>
<td>(\square Q \vdash P \xi \Rightarrow \xi' R)</td>
</tr>
</tbody>
</table>
| VS-ALLOCI | \(\xi \Rightarrow \xi' P\)

**Hoare triples.** The following rules can be derived for Hoare triples.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HT-RET</td>
<td>({\text{True}} w {v, v = w}_{\xi})</td>
</tr>
<tr>
<td>HT-CSQ</td>
<td>(P \Rightarrow P') ({P'} e {v, Q'}_{\xi}) (\forall v. Q' \Rightarrow Q)</td>
</tr>
<tr>
<td>HT-FRAME</td>
<td>({P} e {v, Q}<em>{\xi}) ({P + R} e {v, Q + R}</em>{\xi})</td>
</tr>
<tr>
<td>HT-ATOMIC</td>
<td>(P \xi \Rightarrow \xi' P') ({P'} e {v, Q'}_{\xi'}) (\forall v. Q' \Rightarrow Q') (\xi' \Rightarrow \xi') atomic(e)</td>
</tr>
<tr>
<td>HT-DISJ</td>
<td>({P} e {v, R}<em>{\xi} \quad {Q} e {v, R}</em>{\xi}) ({P \lor Q} e {v, R}_{\xi})</td>
</tr>
<tr>
<td>HT-EXIST</td>
<td>(\forall x. (P) e {v, Q}_{\xi})</td>
</tr>
<tr>
<td>HT-BOX</td>
<td>(\square Q \vdash P \xi \Rightarrow \xi' R)</td>
</tr>
<tr>
<td>HT-INV</td>
<td>(\neg Q \Rightarrow P\xi \Rightarrow \xi' R) (\text{atomic(e)})</td>
</tr>
<tr>
<td>HT-INV-TIMELESS</td>
<td>({R \Rightarrow P} e {v, R \Rightarrow Q}_{\xi}) (\text{atomic(e)}) (\text{timeless(R)})</td>
</tr>
<tr>
<td>HT-FALSE</td>
<td>({\text{False}} e {v, P}_{\xi})</td>
</tr>
<tr>
<td>HT-BOUND</td>
<td>(K) is a context ({P} e {v, Q}<em>{\xi}) (\forall v. {Q} e {v, R}</em>{\xi})</td>
</tr>
<tr>
<td>HT-INV-TIMELESS</td>
<td>({R \Rightarrow P} e {v, R \Rightarrow Q}_{\xi}) (\text{atomic(e)}) (\text{timeless(R)})</td>
</tr>
</tbody>
</table>
Lifting of operational semantics. We can derive some specialized forms of the lifting axioms for the operational semantics.

\[\begin{align*}
\text{wp-lift-atomic-step} & : \\text{atomic}(e_1) \quad \text{red}(e_1, \sigma_1) \\
& \Rightarrow \text{wp}(\sigma_1) \Rightarrow \forall v_2, \sigma_2, e_\ell. (e_1, \sigma_1 \rightarrow \text{val2expr}(v), \sigma_2, e_\ell) \land \text{ Phy}(\sigma_2) \Rightarrow P[v_2/x] \ast \text{ wp} \cdot \ell \left\{ \bot, \text{ True} \right\} \\
\end{align*}\]

\[\begin{align*}
\text{wp-lift-atomic-det-step} & : \\text{atomic}(e_1) \quad \text{red}(e_1, \sigma_1) \\
& \Rightarrow \forall e'_2, \sigma'_2, \ell, e_1, \sigma_1 \rightarrow e_2, \sigma_2, e_\ell \Rightarrow \sigma_2 = \sigma'_2 \land \text{ expr2val}(e'_2) = v_2 \land e_\ell = e'_\ell \\
& \Rightarrow \text{wp}(\sigma_1) \Rightarrow \text{ Phy}(\sigma_2) \Rightarrow P[v_2/x] \ast \text{ wp} \cdot \ell \left\{ \bot, \text{ True} \right\} \\
& \Downarrow \text{ wp} \ell \cdot e_1 \left\{ x, P \right\} \\
\end{align*}\]

\[\begin{align*}
\text{wp-lift-pure-det-step} & : \\text{expr2val}(e_1) = \bot \\
& \forall \sigma_1, \text{red}(e_1, \sigma_1) \\
& \Rightarrow \forall e'_1, e_1, \sigma_1 \rightarrow e_2, \sigma_2, e_\ell \Rightarrow \sigma_1 = \sigma_2 \land e_\ell = e'_\ell \\
& \Rightarrow \text{wp}(\sigma_1) \Rightarrow P[e_1/x] \ast \text{ wp} \cdot \ell \left\{ \bot, \text{ True} \right\} \\
& \Downarrow \text{ wp} \ell \cdot e_1 \left\{ x, P \right\} \\
\end{align*}\]

7.3 Global functor and ghost ownership

Hereinafter we assume the global CMRA functor (served up as a parameter to Iris) is obtained from a family of functors \((\Sigma_i)_{i \in I}\) for some finite \(I\) by picking \(\Sigma(T) \triangleq \prod_{i \in I} \text{GhName} \overset{\text{fin}}{\rightarrow} \Sigma_i(T)\).

We don’t care so much about what concretely \(\text{GhName}\) is, as long as it is countable and infinite. With \(M_i \triangleq \Sigma_i(i\text{Prop})\), we write \(\{i \mapsto [m : M_i]\}\) (or just \([m : M_i]\) if \(M_i\) is clear from the context) for \([\{i \mapsto [m : a]\}\])

In other words, \([i : M_i]\) asserts that in the current state of monoid \(M_i\), the “ghost location” \(\gamma\) is allocated and we own piece \(a\).

From \texttt{pvs-update}, \texttt{vs-update} and the frame-preserving updates in §3.1 and §3.3, we have the following derived rules.

\[\begin{align*}
\text{GHOST-ALLOC-STRONG} & : \ G \text{ infinite} \\
& \Rightarrow \exists \gamma \in G. \{i \mapsto [m : M_i]\} \\
\text{GHOST-ALLOC} & : \ True \Rightarrow \exists \gamma. \{i \mapsto [m : M_i]\} \\
\text{GHOST-UPDATE} & : \ a \Rightarrow_M B \\
& \Rightarrow \exists b \in B. \{i \mapsto [m : M_i]\} \\
\text{GHOST-OP} & : \{i \mapsto [m : M_i]\} \ast \{i \mapsto [n : N_i]\} \Rightarrow \{i \mapsto [m \cdot n : M_i]\} \\
\text{GHOST-VALID} & : \{i \mapsto [m : M_i]\} \Rightarrow V_M(a) \\
\text{GHOST-TIMELESS} & : \ a \text{ is a discrete COFE element} \\
& \Rightarrow \text{timeless}([i \mapsto [m : M_i]]) \\
\end{align*}\]

7.4 Invariant identifier namespaces

Let \(\mathcal{N} \in \text{InvNamesp} \triangleq \text{list(InvName)}\) be the type of names\(\)aces for invariant names. Notice that there is an injection \(\text{namesp}_\text{inj} : \text{InvNamesp} \rightarrow \text{InvName}\). Whenever needed (in particular, for masks at view shifts and Hoare triples), we coerce \(\mathcal{N}\) to its suffix-closure:

\[\mathcal{N} \uparrow \triangleq \{ \ell \mid \exists \mathcal{N}', \ell = \text{namesp}_\text{inj}(\mathcal{N}' + \mathcal{N})\}\]

We use the notation \(\mathcal{N} \uparrow\) for the namespace \([\ell] + \mathcal{N}\).

We define the inclusion relation on namespaces as \(\mathcal{N}_1 \subseteq \mathcal{N}_2 \iff \exists \mathcal{N}_3. \mathcal{N}_2 = \mathcal{N}_3 + \mathcal{N}_1\), i.e., \(\mathcal{N}_1\) is a suffix of \(\mathcal{N}_2\). We have that \(\mathcal{N}_1 \subseteq \mathcal{N}_2 \Rightarrow \mathcal{N}_2 \subseteq \mathcal{N}_1\).

Similarly, we define \(\mathcal{N}_1 \not\subseteq \mathcal{N}_2 \iff \exists \mathcal{N}_3. \mathcal{N}_2 \cap \mathcal{N}_3 \subseteq \mathcal{N}_2 \land |\mathcal{N}_3| = |\mathcal{N}_2| \land \mathcal{N}_1 \not\subseteq \mathcal{N}_2\), i.e., there exists a distinguishing suffix. We have that \(\mathcal{N}_1 \not\subseteq \mathcal{N}_2 \Rightarrow \mathcal{N}_2 \not\subseteq \mathcal{N}_1\), and furthermore \(\ell_1 \neq \ell_2 \Rightarrow \mathcal{N} \uparrow \ell_1 \neq \mathcal{N} \uparrow \ell_2\).
We will overload the usual Iris notation for invariant assertions in the following:

$$\boxed{P}^N \triangleq \exists \iota \in N^\uparrow. \boxed{P}$$

We can now derive the following rules for this derived form of the invariant assertion:

$$\boxed{P}^N \vdash \Box \boxed{P}^N$$

$$\vdash P \vdash \Rightarrow \boxed{P}^N$$

atomic(e)  \hspace{1cm} N \subseteq \mathcal{E} \hspace{1cm} \Theta \vdash \boxed{P}^N \hspace{1cm} \Theta \vdash \vdash P \Rightarrow \text{wp}_{E\setminus N} e \{ v.\vdash P \ast Q \} \hspace{1cm} \Theta \vdash \text{wp}_{E} e \{ v. Q \}

$$N \subseteq \mathcal{E} \hspace{1cm} \Theta \vdash \boxed{P}^N \hspace{1cm} \Theta \vdash \vdash P \Rightarrow \Rightarrow_{E\setminus N} \vdash P \ast Q$$

$$\vdash L \vdash \Rightarrow_{E} Q$$

atomic(e)  \hspace{1cm} N \subseteq \mathcal{E} \hspace{1cm} \{ \vdash P \ast Q \} e \{ v. \vdash P \ast R \}_{E\setminus N} \hspace{1cm} N \subseteq \mathcal{E} \hspace{1cm} \{ \vdash P \ast Q \} \Rightarrow_{E\setminus N} \vdash P \ast R \hspace{1cm} \{ \vdash P \ast Q \} e \{ v. R \}_{E}

$$\boxed{P}^N \vdash \{ Q \} e \{ v. R \}_{E}$$

$$\boxed{P}^N \vdash Q \Rightarrow_{E} R$$
References


