The Iris 3.0 Documentation

http://plv.mpi-sws.org/iris/

November 23, 2017

Abstract

This document describes formally the Iris program logic. Every result in this document has been fully verified in Coq. The latest versions of this document and the Coq formalization can be found in the git repository at https://gitlab.mpi-sws.org/FP/iris-coq/. For further information, visit the Iris project website at http://plv.mpi-sws.org/iris/.

Contents

1 Algebraic Structures **4**
- 1.1 OFE . . . 4
- 1.2 COFE . . . 5
- 1.3 RA . . . 5
- 1.4 CMRA . . . 6

2 OFE and COFE constructions **9**
- 2.1 Trivial pointwise lifting . . . 9
- 2.2 Next (type-level later) . . . 9
- 2.3 Uniform Predicates . . . 9

3 RA and CMRA constructions **10**
- 3.1 Product . . . 10
- 3.2 Sum . . . 10
- 3.3 Option . . . 10
- 3.4 Finite partial function . . . 11
- 3.5 Agreement . . . 11
- 3.6 Exclusive CMRA . . . 11
- 3.7 Authoritative . . . 12
- 3.8 STS with tokens . . . 12

4 Base Logic **15**
- 4.1 Grammar . . . 15
- 4.2 Types . . . 15
- 4.3 Proof rules . . . 16
- 4.4 Consistency . . . 18

5 Model and semantics **19**

6 Extensions of the Base Logic **22**
- 6.1 Derived rules about base connectives . . . 22
- 6.2 Persistent assertions . . . 22
- 6.3 Timeless assertions and except-0 . . . 22

7 Language **24**
- 7.1 Concurrent language . . . 24

8 Program Logic **25**
- 8.1 Dynamic Composeable Higher-Order Resources . . . 25
- 8.2 World Satisfaction, Invariants, Fancy Updates . . . 27
- 8.3 Weakest Precondition . . . 28
- 8.4 Invariant Namespaces . . . 31
- 8.5 Accessors . . . 32

9 Derived constructions **33**
9.1 Non-atomic ("thread-local") invariants 33
9.2 Boxes 34

10 Logical paradoxes **36**
10.1 Saved propositions without a later 36
10.2 Invariants without a later 37

1 Algebraic Structures

1.1 OFE

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

Definition 1. *An* ordered family of equivalences *(OFE) is a tuple* $(T, (\overset{n}{=} \subseteq T \times T)_{n\in\mathbb{N}})$ *satisfying*

$$\forall n.\ (\overset{n}{=}) \text{ is an equivalence relation} \qquad \text{(OFE-EQUIV)}$$

$$\forall n, m.\ n \geq m \Rightarrow (\overset{n}{=}) \subseteq (\overset{m}{=}) \qquad \text{(OFE-MONO)}$$

$$\forall x, y.\ x = y \Leftrightarrow (\forall n.\ x \overset{n}{=} y) \qquad \text{(OFE-LIMIT)}$$

The key intuition behind OFEs is that elements x and y are n-equivalent, notation $x \overset{n}{=} y$, 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, $\overset{n}{=}$ becomes more and more refined (OFE-MONO)—and in the limit, it agrees with plain equality (OFE-LIMIT).

Definition 2. *An element $x \in T$ of an OFE is called* discrete *if*

$$\forall y \in T.\ x \overset{0}{=} y \Rightarrow x = y$$

An OFE A is called discrete *if all its elements are discrete. For a set X, we write ΔX for the discrete OFE with $x \overset{n}{=} x' \triangleq x = x'$*

Definition 3. *A function $f : T \to U$ between two OFEs is* non-expansive *(written $f : T \xrightarrow{ne} U$) if*

$$\forall n, x \in T, y \in T.\ x \overset{n}{=} y \Rightarrow f(x) \overset{n}{=} f(y)$$

It is contractive *if*

$$\forall n, x \in T, y \in T.\ (\forall m < n.\ x \overset{m}{=} y) \Rightarrow f(x) \overset{n}{=} f(y)$$

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.

Definition 4. *The category $\mathcal{OFE}$ consists of OFEs as objects, and non-expansive functions as arrows.*

Note that $\mathcal{OFE}$ is cartesian closed. In particular:

Definition 5. *Given two OFEs T and U, the set of non-expansive functions $\left\{f : T \xrightarrow{ne} U\right\}$ is itself an OFE with*

$$f \overset{n}{=} g \triangleq \forall x \in T.\ f(x) \overset{n}{=} g(x)$$

Definition 6. *A (bi)functor $F : \mathcal{OFE} \to \mathcal{OFE}$ 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 $(-) \xrightarrow{\text{ne}} (-)$ 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.

1.2 COFE

COFEs are *complete OFEs*, which means that we can take limits of arbitrary chains.

Definition 7 (Chain)**.** *Given some set T and an indexed family $(\overset{n}{=} \subseteq T \times T)_{n\in\mathbb{N}}$ of equivalence relations, a* chain *is a function $c : \mathbb{N} \to T$ such that $\forall n, m.\, n \leq m \Rightarrow c(m) \overset{n}{=} c(n)$.*

Definition 8. *A* complete ordered family of equivalences *(COFE) is a tuple $(T : \mathcal{OFE}, \lim : \text{chain}(T) \to T)$ satisfying*

$$\forall n, c.\ \lim(c) \overset{n}{=} c(n) \qquad \text{(COFE-COMPL)}$$

Definition 9. *The category $\mathcal{COFE}$ consists of COFEs as objects, and non-expansive functions as arrows.*

The function space $T \xrightarrow{\text{ne}} U$ is a COFE if U is a COFE (*i.e.,* the domain T can actually be just an OFE).

Completeness is necessary to take fixed-points. For once, every *contractive function* $f : T \to U$ where U is a COFE and inhabited has a *unique* fixed-point $\mathit{fix}(f)$ such that $\mathit{fix}(f) = f(\mathit{fix}(f))$. This also holds if f^k is contractive for an arbitrary k. Furthermore, by America and Rutten's theorem [1, 3], every contractive (bi)functor from $\mathcal{COFE}$ to $\mathcal{COFE}$ has a unique[1] fixed-point.

1.3 RA

Definition 10. *A* resource algebra *(RA) is a tuple $(M, \mathcal{V} \subseteq M, |-| : M \to M^?, (\cdot) : M \times M \to M)$ satisfying:*

$$\forall a, b, c.\ (a \cdot b) \cdot c = a \cdot (b \cdot c) \qquad \text{(RA-ASSOC)}$$
$$\forall a, b.\ a \cdot b = b \cdot a \qquad \text{(RA-COMM)}$$
$$\forall a.\ |a| \in M \Rightarrow |a| \cdot a = a \qquad \text{(RA-CORE-ID)}$$
$$\forall a.\ |a| \in M \Rightarrow ||a|| = |a| \qquad \text{(RA-CORE-IDEM)}$$
$$\forall a, b.\ |a| \in M \land a \preccurlyeq b \Rightarrow |b| \in M \land |a| \preccurlyeq |b| \qquad \text{(RA-CORE-MONO)}$$
$$\forall a, b.\ (a \cdot b) \in \mathcal{V} \Rightarrow a \in \mathcal{V} \qquad \text{(RA-VALID-OP)}$$

where $M^? \triangleq M \uplus \{\bot\}$ $\qquad a^? \cdot \bot \triangleq \bot \cdot a^? \triangleq a^?$

$$a \preccurlyeq b \triangleq \exists c \in M.\, b = a \cdot c \qquad \text{(RA-INCL)}$$

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 $\mathcal{V}$ of *valid* elements that is compatible with the composition operation (RA-VALID-OP).

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

[1] Uniqueness is not proven in Coq.

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

   Notice that the domain of the core is $M^?$, a set that adds a dummy element $\bot$ to M. Thus, the core can be *partial*: not all elements need to have a unit. We use the metavariable $a^?$ to indicate elements of $M^?$. We also lift the composition $(\cdot)$ to $M^?$. 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 $|-|$ can always be picked as a constant function.

Definition 11. *It is possible to do a* frame-preserving update *from $a \in M$ to $B \subseteq M$, written $a \rightsquigarrow B$, if*

$$\forall a_{\mathrm{f}}^? \in M^?.\, a \cdot a_{\mathrm{f}}^? \in \mathcal{V} \Rightarrow \exists b \in B.\, b \cdot a_{\mathrm{f}}^? \in \mathcal{V}$$

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

The assertion $a \rightsquigarrow B$ says that every element $a_{\mathrm{f}}^?$ compatible with a (we also call such elements *frames*), must also be compatible with some $b \in B$. Notice that $a_{\mathrm{f}}^?$ could be $\bot$, 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 γ, 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 $b \in B$.

1.4 CMRA

Definition 12. *A* CMRA *is a tuple* $(M : \mathcal{OFE}, (\mathcal{V}_n \subseteq M)_{n\in\mathbb{N}}, |-| : M \xrightarrow{ne} M^?, (\cdot) : M \times M \xrightarrow{ne} M)$ *satisfying:*

$$\forall n, a, b.\, a \overset{n}{=} b \land a \in \mathcal{V}_n \Rightarrow b \in \mathcal{V}_n \qquad \text{(CMRA-VALID-NE)}$$

$$\forall n, m.\, n \geq m \Rightarrow \mathcal{V}_n \subseteq \mathcal{V}_m \qquad \text{(CMRA-VALID-MONO)}$$

$$\forall a, b, c.\, (a \cdot b) \cdot c = a \cdot (b \cdot c) \qquad \text{(CMRA-ASSOC)}$$

$$\forall a, b.\, a \cdot b = b \cdot a \qquad \text{(CMRA-COMM)}$$

$$\forall a.\, |a| \in M \Rightarrow |a| \cdot a = a \qquad \text{(CMRA-CORE-ID)}$$

$$\forall a.\, |a| \in M \Rightarrow ||a|| = |a| \qquad \text{(CMRA-CORE-IDEM)}$$

$$\forall a, b.\, |a| \in M \land a \preccurlyeq b \Rightarrow |b| \in M \land |a| \preccurlyeq |b| \qquad \text{(CMRA-CORE-MONO)}$$

$$\forall n, a, b.\, (a \cdot b) \in \mathcal{V}_n \Rightarrow a \in \mathcal{V}_n \qquad \text{(CMRA-VALID-OP)}$$

$$\forall n, a, b_1, b_2.\, a \in \mathcal{V}_n \land a \overset{n}{=} b_1 \cdot b_2 \Rightarrow \exists c_1, c_2.\, a = c_1 \cdot c_2 \land c_1 \overset{n}{=} b_1 \land c_2 \overset{n}{=} b_2 \qquad \text{(CMRA-EXTEND)}$$

where

$$a \preccurlyeq b \triangleq \exists c.\, b = a \cdot c \qquad \text{(CMRA-INCL)}$$

$$a \overset{n}{\preccurlyeq} b \triangleq \exists c.\, b \overset{n}{=} a \cdot c \qquad \text{(CMRA-INCLN)}$$

This is a natural generalization of RAs over OFEs. 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:

$$\begin{array}{ccc} a & \overset{n}{=} & b \\ \| & & \| \\ a_1 \cdot a_2 & \overset{n}{=} & b_1 \cdot b_2 \end{array}$$

where the n-equivalence at the bottom is meant to apply to the pairs of elements, *i.e.,* we demand $a_1 \overset{n}{=} b_1$ and $a_2 \overset{n}{=} b_2$. 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:

$$\triangleright(P * Q) \Leftrightarrow \triangleright P * \triangleright Q$$

Definition 13. *An element ε of a CMRA M is called the* unit *of M if it satisfies the following conditions:*

1. *ε is valid:*
   $\forall n.\, \varepsilon \in \mathcal{V}_n$
2. *ε is a left-identity of the operation:*
   $\forall a \in M.\, \varepsilon \cdot a = a$
3. *ε is its own core:*
   $|\varepsilon| = \varepsilon$

Lemma 1. *If M has a unit ε, then the core $|-|$ is total,* i.e., $\forall a.\, |a| \in M$.

Definition 14. *It is possible to do a* frame-preserving update *from $a \in M$ to $B \subseteq M$, written $a \rightsquigarrow B$, if*

$$\forall n, a_{\mathrm{f}}^{?}.\, a \cdot a_{\mathrm{f}}^{?} \in \mathcal{V}_n \Rightarrow \exists b \in B.\, b \cdot a_{\mathrm{f}}^{?} \in \mathcal{V}_n$$

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 15. *A CMRA M is* discrete *if it satisfies the following conditions:*

1. *M is a discrete COFE*
2. *$\mathcal{V}$ ignores the step-index:*
   $\forall a \in M.\, a \in \mathcal{V}_0 \Rightarrow \forall n, a \in \mathcal{V}_n$

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 $\mathcal{V}$.

Definition 16. *A function $f : M_1 \to M_2$ between two CMRAs is* monotone *(written $f : M_1 \xrightarrow{mon} M_2$) if it satisfies the following conditions:*

1. *f is non-expansive*
2. *f preserves validity:*
   $\forall n, a \in M_1.\, a \in \mathcal{V}_n \Rightarrow f(a) \in \mathcal{V}_n$
3. *f preserves CMRA inclusion:*
   $\forall a \in M_1, b \in M_1.\, a \preccurlyeq b \Rightarrow f(a) \preccurlyeq f(b)$

Definition 17. *The category $\mathcal{CMRA}$ consists of CMRAs as objects, and monotone functions as arrows.*

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

2 OFE and COFE constructions

2.1 Trivial pointwise lifting

The (C)OFE structure on many types can be easily obtained by pointwise lifting of the structure of the components. This is what we do for option $T^?$, product $(M_i)_{i\in I}$ (with I some finite index set), sum $T + T'$ and finite partial functions $K \xrightarrow{\text{fin}} M$ (with K infinite countable).

2.2 Next (type-level later)

Given a OFE T, we define $\blacktriangleright T$ as follows (using a datatype-like notation to define the type):

$$\blacktriangleright T \triangleq \mathsf{next}(x : T)$$
$$\mathsf{next}(x) \overset{n}{=} \mathsf{next}(y) \triangleq n = 0 \vee x \overset{n-1}{=} y$$

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

$\blacktriangleright(-)$ is a locally *contractive* functor from $\mathcal{OFE}$ to $\mathcal{OFE}$.

2.3 Uniform Predicates

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

$$\mathit{UPred}(M) \triangleq \left\{ \Phi : \mathbb{N} \times M \to \mathit{Prop} \;\middle|\; \begin{array}{l} (\forall n, x, y.\, \Phi(n, x) \wedge x \overset{n}{=} y \Rightarrow \Phi(n, y)) \wedge \\ (\forall n, m, x, y.\, \Phi(n, x) \wedge x \preccurlyeq y \wedge m \leq n \wedge y \in \mathcal{V}_m \Rightarrow \Phi(m, y)) \end{array} \right\}$$

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

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.

$$\begin{aligned} \mathit{SProp} &\triangleq \wp^{\downarrow}(\mathbb{N}) \\ &\triangleq \{X \in \wp(\mathbb{N}) \mid \forall n, m.\, n \geq m \Rightarrow n \in X \Rightarrow m \in X\} \\ X \overset{n}{=} Y &\triangleq \forall m \leq n.\, m \in X \Leftrightarrow m \in Y \end{aligned}$$

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

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

$$\begin{aligned} \mathit{UPred}(M) &\cong M \xrightarrow{\text{mon}} \mathit{SProp} \\ &\triangleq \left\{ \Phi : M \xrightarrow{\text{ne}} \mathit{SProp} \;\middle|\; \forall n, m, x, y.\, n \in \Phi(x) \wedge x \preccurlyeq y \wedge m \leq n \wedge y \in \mathcal{V}_m \Rightarrow m \in \Phi(y) \right\} \end{aligned}$$

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 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:

$$\text{PROD-UPDATE}\quad \frac{a \rightsquigarrow_{M_i} B}{f\,[i \leftarrow a] \rightsquigarrow \{f\,[i \leftarrow b] \mid b \in B\}}$$

3.2 Sum

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

$$\begin{aligned}
M_1 +_\lightning M_2 &\triangleq \mathsf{inl}(a_1 : M_1) \mid \mathsf{inr}(a_2 : M_2) \mid \lightning \\
\mathcal{V}_n &\triangleq \{\mathsf{inl}(a_1) \mid a_1 \in \mathcal{V}'_n\} \cup \{\mathsf{inr}(a_2) \mid a_2 \in \mathcal{V}''_n\} \\
\mathsf{inl}(a_1) \cdot \mathsf{inl}(b_1) &\triangleq \mathsf{inl}(a_1 \cdot b_1) \\
|\mathsf{inl}(a_1)| &\triangleq \begin{cases} \bot & \text{if } |a_1| = \bot \\ \mathsf{inl}(|a_1|) & \text{otherwise} \end{cases}
\end{aligned}$$

The composition and core for inr are defined symmetrically. The remaining cases of the composition and core are all $\lightning$. Above, $\mathcal{V}'$ refers to the validity of M_1, and $\mathcal{V}''$ to the validity of M_2.

Notice that we added the artificial "invalid" (or "undefined") element $\lightning$ to this CMRA just in order to make certain compositions of elements (in this case, inl and inr) invalid.

The step-indexed equivalence is inductively defined as follows:

$$\frac{x \overset{n}{=} y}{\mathsf{inl}(x) \overset{n}{=} \mathsf{inl}(y)} \qquad \frac{x \overset{n}{=} y}{\mathsf{inr}(x) \overset{n}{=} \mathsf{inr}(y)} \qquad \lightning \overset{n}{=} \lightning$$

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

$$\text{SUM-UPDATE}\quad \frac{a \rightsquigarrow_{M_1} B}{\mathsf{inl}(a) \rightsquigarrow \{\mathsf{inl}(b) \mid b \in B\}} \qquad \text{SUM-SWAP}\quad \frac{\forall a_\mathrm{f}, n.\, a \cdot a_\mathrm{f} \notin \mathcal{V}'_n \qquad b \in \mathcal{V}''}{\mathsf{inl}(a) \rightsquigarrow \mathsf{inr}(b)}$$

Crucially, the second rule allows us to *swap* the "side" of the sum that the CMRA is on if $\mathcal{V}$ has *no possible frame*.

3.3 Option

The definition of the (CM)RA axioms already lifted the composition operation on M to one on $M^?$. We can easily extend this to a full CMRA by defining a suitable core, namely

$$\begin{aligned}
|\bot| &\triangleq \bot \\
|a^?| &\triangleq |a| \qquad \text{If } a^? \neq \bot
\end{aligned}$$

Notice that this core is total, as the result always lies in $M^?$ (rather than in $M^{??}$).

3.4 Finite partial function

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

We obtain the following frame-preserving updates:

FPFN-ALLOC-STRONG
$$\frac{G \text{ infinite} \qquad a \in \mathcal{V}}{\emptyset \rightsquigarrow \{[\gamma \leftarrow a] \mid \gamma \in G\}}$$

FPFN-ALLOC
$$\frac{a \in \mathcal{V}}{\emptyset \rightsquigarrow \{[\gamma \leftarrow a] \mid \gamma \in K\}}$$

FPFN-UPDATE
$$\frac{a \rightsquigarrow_M B}{f\,[i \leftarrow a]] \rightsquigarrow \{f\,[i \leftarrow b] \mid b \in B\}}$$

Above, $\mathcal{V}$ refers to the validity of M.

$K \xrightarrow{\text{fin}} (-)$ is a locally non-expansive functor from $\mathcal{CMRA}$ to $\mathcal{CMRA}$.

3.5 Agreement

Given some OFE T, we define the CMRA $\textsc{Ag}(T)$ as follows:

$$\begin{aligned}
\textsc{Ag}(T) &\triangleq \left\{a \in \wp^{\text{fin}}(T) \,\middle|\, a \neq \emptyset\right\} / \sim \\
a \overset{n}{=} b &\triangleq (\forall x \in a.\, \exists y \in b.\, x \overset{n}{=} y) \wedge (\forall y \in b.\, \exists x \in a.\, x \overset{n}{=} y) \\
&\text{where } a \sim b \triangleq \forall n.\, a \overset{n}{=} b \\
\\
\mathcal{V}_n &\triangleq \left\{a \in \textsc{Ag}(T) \,\middle|\, \forall x, y \in a.\, x \overset{n}{=} y\right\} \\
|a| &\triangleq a \\
a \cdot b &\triangleq a \cup b
\end{aligned}$$

$\textsc{Ag}(-)$ is a locally non-expansive functor from $\mathcal{OFE}$ to $\mathcal{CMRA}$.

We define a non-expansive injection ag into $\textsc{Ag}(T)$ as follows:

$$\mathsf{ag}(x) \triangleq \{x\}$$

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

AG-VAL
$$\mathsf{ag}(x) \in \mathcal{V}_n$$

AG-DUP
$$\mathsf{ag}(x) = \mathsf{ag}(x) \cdot \mathsf{ag}(x)$$

AG-AGREE
$$\mathsf{ag}(x) \cdot \mathsf{ag}(y) \in \mathcal{V}_n \Leftrightarrow x \overset{n}{=} y$$

3.6 Exclusive CMRA

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

$$\begin{aligned}
\textsc{Ex}(T) &\triangleq \mathsf{ex}(T) \mid \lightning \\
\mathcal{V}_n &\triangleq \{a \in \textsc{Ex}(T) \mid a \neq \lightning\}
\end{aligned}$$

All cases of composition go to $\lightning$.

$$|\mathsf{ex}(x)| \triangleq \bot \qquad\qquad |\lightning| \triangleq \lightning$$

Remember that $\bot$ is the "dummy" element in $M^?$ indicating (in this case) that $\mathsf{ex}(x)$ has no core.

The step-indexed equivalence is inductively defined as follows:

$$\frac{x \stackrel{n}{=} y}{\mathsf{ex}(x) \stackrel{n}{=} \mathsf{ex}(y)} \qquad \lightning \stackrel{n}{=} \lightning$$

$\textsc{Ex}(-)$ is a locally non-expansive functor from $\mathcal{OFE}$ to $\mathcal{CMRA}$.

We obtain the following frame-preserving update:

$$\frac{}{\mathsf{ex}(x) \rightsquigarrow \mathsf{ex}(y)}\ \textsc{ex-update}$$

3.7 Authoritative

Given a CMRA M, we construct $\textsc{Auth}(M)$ modeling someone owning an *authoritative* element a of M, and others potentially owning fragments $b \preccurlyeq a$ of a. We assume that M has a unit ε, and hence its core is total. (If M is an exclusive monoid, the construction is very similar to a half-ownership monoid with two asymmetric halves.)

$$\begin{aligned}
\textsc{Auth}(M) &\triangleq \textsc{Ex}(M)^? \times M \\
\mathcal{V}_n &\triangleq \{(x, b) \in \textsc{Auth}(M) \mid b \in \mathcal{V}_n \wedge (x = \bot \vee \exists a.\, x = \mathsf{ex}(a) \wedge b \preccurlyeq_n a)\} \\
(x_1, b_1) \cdot (x_2, b_2) &\triangleq (x_1 \cdot x_2, b_2 \cdot b_2) \\
|(x, b)| &\triangleq (\bot, |b|) \\
(x_1, b_1) \stackrel{n}{=} (x_2, b_2) &\triangleq x_1 \stackrel{n}{=} x_2 \wedge b_1 \stackrel{n}{=} b_2
\end{aligned}$$

Note that $(\bot, \varepsilon)$ is the unit and asserts no ownership whatsoever, but $(\mathsf{ex}(\varepsilon), \varepsilon)$ asserts that the authoritative element is ε.

Let $a, b \in M$. We write $\bullet a$ for full ownership $(\mathsf{ex}(a), \varepsilon)$ and $\circ b$ for fragmental ownership $(\bot, b)$ and $\bullet a, \circ b$ for combined ownership $(\mathsf{ex}(a), b)$.

The frame-preserving update involves the notion of a *local update*:

Definition 18. *It is possible to do a* local update *from a_1 and b_1 to a_2 and b_2, written $(a_1, b_1) \overset{\mathrm{l}}{\rightsquigarrow} (a_2, b_2)$, if*

$$\forall n, a_{\mathrm{f}}^?.\, a_1 \in \mathcal{V}_n \wedge a_1 \stackrel{n}{=} b_1 \cdot a_{\mathrm{f}}^? \Rightarrow a_2 \in \mathcal{V}_n \wedge a_2 \stackrel{n}{=} b_2 \cdot a_{\mathrm{f}}^?$$

In other words, the idea is that for every possible frame $a_{\mathrm{f}}^?$ completing b_1 to a_1, the same frame also completes b_2 to a_2.

We then obtain

$$\frac{(a_1, b_1) \overset{\mathrm{l}}{\rightsquigarrow} (a_2, b_2)}{\bullet a_1, \circ b_1 \rightsquigarrow \bullet a_2, \circ b_2}\ \textsc{auth-update}$$

3.8 STS with tokens

Given a state-transition system (STS, *i.e.,* a directed graph) $(\mathcal{S}, \to \subseteq \mathcal{S} \times \mathcal{S})$, a set of tokens $\mathcal{T}$, and a labeling $\mathcal{L} : \mathcal{S} \to \wp(\mathcal{T})$ 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 [6]. We first lift the transition relation to $\mathcal{S} \times \wp(\mathcal{T})$ (implementing a *law of token conservation*) and define a stepping relation for the *frame* of a given token set:

$$(s, T) \to (s', T') \triangleq s \to s' \land \mathcal{L}(s) \uplus T = \mathcal{L}(s') \uplus T'$$
$$s \xrightarrow{T} s' \triangleq \exists T_1, T_2.\, T_1 \mathbin{\#} \mathcal{L}(s) \cup T \land (s, T_1) \to (s', T_2)$$

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

$$\mathsf{closed}(S, T) \triangleq \forall s \in S.\, \mathcal{L}(s) \mathbin{\#} T \land \left(\forall s'.\, s \xrightarrow{T} s' \Rightarrow s' \in S\right)$$
$$\uparrow(S, T) \triangleq \left\{ s' \in \mathcal{S} \,\middle|\, \exists s \in S.\, s \xrightarrow{T}{}^{*}\, s' \right\}$$

The STS RA is defined as follows

$$\begin{aligned}
M &\triangleq \mathsf{auth}(s : \mathcal{S}, T : \wp(\mathcal{T}) \mid \mathcal{L}(s) \mathbin{\#} T) \mid \\
&\qquad \mathsf{frag}(S : \wp(\mathcal{S}), T : \wp(\mathcal{T}) \mid \mathsf{closed}(S, T) \land S \neq \emptyset) \mid \lightning \\
\mathcal{V} &\triangleq \{a \in M \mid a \neq \lightning\} \\
\mathsf{frag}(S_1, T_1) \cdot \mathsf{frag}(S_2, T_2) &\triangleq \mathsf{frag}(S_1 \cap S_2, T_1 \cup T_2) \qquad \text{if } T_1 \mathbin{\#} T_2 \text{ and } S_1 \cap S_2 \neq \emptyset \\
\mathsf{frag}(S, T) \cdot \mathsf{auth}(s, T') &\triangleq \mathsf{auth}(s, T') \cdot \mathsf{frag}(S, T) \triangleq \mathsf{auth}(s, T \cup T') \qquad \text{if } T \mathbin{\#} T' \text{ and } s \in S \\
|\mathsf{frag}(S, T)| &\triangleq \mathsf{frag}(\uparrow(S, \emptyset), \emptyset) \\
|\mathsf{auth}(s, T)| &\triangleq \mathsf{frag}(\uparrow(\{s\}, \emptyset), \emptyset)
\end{aligned}$$

The remaining cases are all $\lightning$.

We will need the following frame-preserving update:

$$\text{STS-STEP}\ \frac{(s, T) \to^{*} (s', T')}{\mathsf{auth}(s, T) \rightsquigarrow \mathsf{auth}(s', T')} \qquad \text{STS-WEAKEN}\ \frac{\mathsf{closed}(S_2, T_2) \qquad S_1 \subseteq S_2 \qquad T_2 \subseteq T_1}{\mathsf{frag}(S_1, T_1) \rightsquigarrow \mathsf{frag}(S_2, T_2)}$$

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:

$$a \triangleq \mathsf{frag}(\{\mathsf{s}_1, \mathsf{s}_2\}, \{\mathrm{T}_1\}) \qquad b \triangleq \mathsf{frag}(\{\mathsf{s}_1, \mathsf{s}_3\}, \{\mathrm{T}_2\})$$

We have:

$$a \cdot b = \mathsf{frag}(\{\mathsf{s}_1\}, \{\mathrm{T}_1, \mathrm{T}_2\}) \qquad |a| = \mathsf{frag}(\{\mathsf{s}_1, \mathsf{s}_2, \mathsf{s}_4\}, \emptyset) \qquad |b| = \mathsf{frag}(\{\mathsf{s}_1, \mathsf{s}_3, \mathsf{s}_4\}, \emptyset)$$

$$|a| \cdot |b| = \mathsf{frag}(\{\mathsf{s}_1, \mathsf{s}_4\}, \emptyset) \neq |a \cdot b| = \mathsf{frag}(\{\mathsf{s}_1\}, \emptyset)$$

4 Base Logic

The base logic is parameterized by an arbitrary CMRA M having a unit ε. By Lemma 1, this means that the core of M is a total function, so we will treat it as such in the following. This defines the structure of resources that can be owned.

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

$$\mathcal{T} \supseteq \{\mathsf{M}, \mathsf{iProp}\}$$

Elements of $\mathcal{T}$ are ranged over by T.

Each function symbol in $\mathcal{F}$ has an associated *arity* comprising a natural number n and an ordered list of $n+1$ types τ (the grammar of τ is defined below, and depends only on $\mathcal{T}$). We write

$$F : \tau_1, \ldots, \tau_n \to \tau_{n+1} \in \mathcal{F}$$

to express that F is a function symbol with the indicated arity.

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

4.1 Grammar

Syntax. Iris syntax is built up from a signature $\mathcal{S}$ and a countably infinite set *Var* of variables (ranged over by metavariables x, y, z). Below, a ranges over M and i ranges over $\{1, 2\}$.

$$\begin{aligned}
\tau &::= T \mid 1 \mid \tau \times \tau \mid \tau \to \tau \\
t, P, \Phi &::= x \mid F(t_1, \ldots, t_n) \mid () \mid (t, t) \mid \pi_i\ t \mid \lambda x : \tau.\, t \mid t(t) \mid a \mid |t| \mid t \cdot t \mid \\
&\quad \mathsf{False} \mid \mathsf{True} \mid t =_\tau t \mid P \Rightarrow P \mid P \wedge P \mid P \vee P \mid P * P \mid P \mathrel{-\!\!*} P \mid \\
&\quad \mu\, x : \tau.\, t \mid \exists x : \tau.\, P \mid \forall x : \tau.\, P \mid \mathsf{Own}\,(t) \mid \mathcal{V}(t) \mid \Box P \mid \triangleright P \mid \Rrightarrow P
\end{aligned}$$

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

Note that the modalities $\Rrightarrow$, $\Box$ and $\triangleright$ bind more tightly than $*$, $\mathrel{-\!\!*}$, $\wedge$, $\vee$, and $\Rightarrow$.

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.

4.2 Types

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

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 Γ.

Well-typed terms $\boxed{\Gamma \vdash_{\mathcal{S}} t : \tau}$

$$x : \tau \vdash x : \tau \qquad \frac{\Gamma \vdash t : \tau}{\Gamma, x : \tau' \vdash t : \tau} \qquad \frac{\Gamma, x : \tau', y : \tau' \vdash t : \tau}{\Gamma, x : \tau' \vdash t[x/y] : \tau} \qquad \frac{\Gamma_1, x : \tau', y : \tau'', \Gamma_2 \vdash t : \tau}{\Gamma_1, x : \tau'', y : \tau', \Gamma_2 \vdash t[y/x, x/y] : \tau}$$

$$\frac{\Gamma \vdash t_1 : \tau_1 \qquad \cdots \qquad \Gamma \vdash t_n : \tau_n \qquad F : \tau_1, \ldots, \tau_n \to \tau_{n+1} \in \mathcal{F}}{\Gamma \vdash F(t_1, \ldots, t_n) : \tau_{n+1}} \qquad \Gamma \vdash () : 1$$

$$\frac{\Gamma \vdash t : \tau_1 \qquad \Gamma \vdash u : \tau_2}{\Gamma \vdash (t, u) : \tau_1 \times \tau_2} \qquad \frac{\Gamma \vdash t : \tau_1 \times \tau_2 \qquad i \in \{1, 2\}}{\Gamma \vdash \pi_i\, t : \tau_i} \qquad \frac{\Gamma, x : \tau \vdash t : \tau'}{\Gamma \vdash \lambda x.\, t : \tau \to \tau'}$$

$$\frac{\Gamma \vdash t : \tau \to \tau' \qquad u : \tau}{\Gamma \vdash t(u) : \tau'} \qquad \Gamma \vdash \varepsilon : \mathsf{M} \qquad \frac{\Gamma \vdash a : \mathsf{M}}{\Gamma \vdash |a| : \mathsf{M}} \qquad \frac{\Gamma \vdash a : \mathsf{M} \qquad \Gamma \vdash b : \mathsf{M}}{\Gamma \vdash a \cdot b : \mathsf{M}}$$

$$\Gamma \vdash \mathsf{False} : \mathsf{iProp} \qquad \Gamma \vdash \mathsf{True} : \mathsf{iProp} \qquad \frac{\Gamma \vdash t : \tau \qquad \Gamma \vdash u : \tau}{\Gamma \vdash t =_\tau u : \mathsf{iProp}} \qquad \frac{\Gamma \vdash P : \mathsf{iProp} \qquad \Gamma \vdash Q : \mathsf{iProp}}{\Gamma \vdash P \Rightarrow Q : \mathsf{iProp}}$$

$$\frac{\Gamma \vdash P : \mathsf{iProp} \qquad \Gamma \vdash Q : \mathsf{iProp}}{\Gamma \vdash P \wedge Q : \mathsf{iProp}} \qquad \frac{\Gamma \vdash P : \mathsf{iProp} \qquad \Gamma \vdash Q : \mathsf{iProp}}{\Gamma \vdash P \vee Q : \mathsf{iProp}}$$

$$\frac{\Gamma \vdash P : \mathsf{iProp} \qquad \Gamma \vdash Q : \mathsf{iProp}}{\Gamma \vdash P * Q : \mathsf{iProp}} \qquad \frac{\Gamma \vdash P : \mathsf{iProp} \qquad \Gamma \vdash Q : \mathsf{iProp}}{\Gamma \vdash P \mathrel{-\!\!*} Q : \mathsf{iProp}}$$

$$\frac{\Gamma, x : \tau \vdash t : \tau \qquad x \text{ is guarded in } t}{\Gamma \vdash \mu\, x : \tau.\, t : \tau} \qquad \frac{\Gamma, x : \tau \vdash P : \mathsf{iProp}}{\Gamma \vdash \exists x : \tau.\, P : \mathsf{iProp}} \qquad \frac{\Gamma, x : \tau \vdash P : \mathsf{iProp}}{\Gamma \vdash \forall x : \tau.\, P : \mathsf{iProp}}$$

$$\frac{\Gamma \vdash a : \mathsf{M}}{\Gamma \vdash \mathsf{Own}\,(a) : \mathsf{iProp}} \qquad \frac{\Gamma \vdash a : \tau \qquad \tau \text{ is a CMRA}}{\Gamma \vdash \mathcal{V}(a) : \mathsf{iProp}} \qquad \frac{\Gamma \vdash P : \mathsf{iProp}}{\Gamma \vdash \Box P : \mathsf{iProp}} \qquad \frac{\Gamma \vdash P : \mathsf{iProp}}{\Gamma \vdash \triangleright P : \mathsf{iProp}}$$

$$\frac{\Gamma \vdash P : \mathsf{iProp}}{\Gamma \vdash \Rrightarrow P : \mathsf{iProp}}$$

4.3 Proof rules

The judgment $\Gamma \mid P \vdash Q$ says that with free variables Γ, proposition Q holds whenever assumption P holds. Most of the rules will entirely omit the variable contexts Γ. In this case, we assume the same arbitrary context is used for every constituent of the rules. Axioms $\Gamma \mid P \dashv\vdash Q$ indicate that both $\Gamma \mid P \vdash Q$ and $\Gamma \mid Q \vdash P$ are proof rules of the logic.

$\boxed{\Gamma \mid P \vdash Q}$

Laws of intuitionistic higher-order logic with equality. This is entirely standard.

$$\textsc{Asm}\ \frac{}{P \vdash P} \qquad \textsc{Cut}\ \frac{P \vdash Q \qquad Q \vdash R}{P \vdash R} \qquad \textsc{Eq}\ \frac{\Gamma, x : \tau \vdash Q : \mathsf{iProp} \qquad \Gamma \mid P \vdash Q[t/x] \qquad \Gamma \mid P \vdash t =_\tau t'}{\Gamma \mid P \vdash Q[t'/x]}$$

$$\textsc{Refl}\ \frac{}{\mathsf{True} \vdash t =_\tau t} \qquad \bot\textsc{E}\ \frac{}{\mathsf{False} \vdash P} \qquad \top\textsc{I}\ \frac{}{P \vdash \mathsf{True}} \qquad \wedge\textsc{I}\ \frac{P \vdash Q \qquad P \vdash R}{P \vdash Q \wedge R} \qquad \wedge\textsc{EL}\ \frac{P \vdash Q \wedge R}{P \vdash Q} \qquad \wedge\textsc{ER}\ \frac{P \vdash Q \wedge R}{P \vdash R}$$

$$\vee\textsc{IL}\ \frac{P \vdash Q}{P \vdash Q \vee R} \qquad \vee\textsc{IR}\ \frac{P \vdash R}{P \vdash Q \vee R} \qquad \vee\textsc{E}\ \frac{P \vdash R \qquad Q \vdash R}{P \vee Q \vdash R} \qquad \Rightarrow\textsc{I}\ \frac{P \wedge Q \vdash R}{P \vdash Q \Rightarrow R} \qquad \Rightarrow\textsc{E}\ \frac{P \vdash Q \Rightarrow R \qquad P \vdash Q}{P \vdash R}$$

$$\forall\textsc{I}\ \frac{\Gamma, x : \tau \mid P \vdash Q}{\Gamma \mid P \vdash \forall x : \tau. Q} \qquad \forall\textsc{E}\ \frac{\Gamma \mid P \vdash \forall x : \tau. Q \qquad \Gamma \vdash t : \tau}{\Gamma \mid P \vdash Q[t/x]}$$

$$\exists\textsc{I}\ \frac{\Gamma \mid P \vdash Q[t/x] \qquad \Gamma \vdash t : \tau}{\Gamma \mid P \vdash \exists x : \tau. Q} \qquad \exists\textsc{E}\ \frac{\Gamma, x : \tau \mid P \vdash Q}{\Gamma \mid \exists x : \tau. P \vdash Q}$$

Furthermore, we have the usual η and β laws for projections, λ and μ.

Laws of (affine) bunched implications.

$$\begin{array}{c} \mathsf{True} * P \dashv\vdash P \\ P * Q \ \vdash\ Q * P \\ (P * Q) * R \ \vdash\ P * (Q * R) \end{array} \qquad *\textsc{-mono}\ \frac{P_1 \vdash Q_1 \qquad P_2 \vdash Q_2}{P_1 * P_2 \vdash Q_1 * Q_2} \qquad -\!\!*\textsc{I-E}\ \frac{P * Q \vdash R}{P \vdash Q -\!\!* R}$$

Laws for the always modality.

$$\Box\textsc{-mono}\ \frac{P \vdash Q}{\Box P \vdash \Box Q} \qquad \Box\textsc{-E}\ \Box P \vdash P$$

$$\begin{array}{c} \mathsf{True} \vdash \Box\, \mathsf{True} \\ \Box\,(P \wedge Q) \vdash \Box\,(P * Q) \\ \Box P \wedge Q \vdash \Box P * Q \end{array} \qquad \begin{array}{c} \Box P \vdash \Box\Box P \\ \forall x.\ \Box P \vdash \Box \forall x.\ P \\ \Box \exists x.\ P \vdash \exists x.\ \Box P \end{array}$$

Laws for the later modality.

$$\triangleright\textsc{-mono}\ \frac{P \vdash Q}{\triangleright P \vdash \triangleright Q} \qquad \textsc{Löb}\ (\triangleright P \Rightarrow P) \vdash P$$

$$\begin{array}{l} \forall x.\ \triangleright P \vdash \triangleright \forall x.\ P \\ \triangleright \exists x.\ P \vdash \triangleright\, \mathsf{False} \vee \exists x.\ \triangleright P \\ \qquad \triangleright P \vdash \triangleright\, \mathsf{False} \vee (\triangleright\, \mathsf{False} \Rightarrow P) \end{array} \qquad \begin{array}{c} \triangleright\,(P * Q) \dashv\vdash \triangleright P * \triangleright Q \\ \Box \triangleright P \dashv\vdash \triangleright \Box P \end{array}$$

Laws for resources and validity.

$$\begin{aligned} \mathsf{Own}\,(a) * \mathsf{Own}\,(b) &\dashv\vdash \mathsf{Own}\,(a \cdot b) \\ \mathsf{Own}\,(a) &\vdash \Box\,\mathsf{Own}\,(|a|) \\ \mathsf{True} &\vdash \mathsf{Own}\,(\varepsilon) \\ \triangleright\,\mathsf{Own}\,(a) &\vdash \exists b.\, \mathsf{Own}\,(b) \wedge \triangleright(a = b) \end{aligned} \qquad \begin{aligned} \mathsf{Own}\,(a) &\vdash \mathcal{V}(a) \\ \mathcal{V}(a \cdot b) &\vdash \mathcal{V}(a) \\ \mathcal{V}(a) &\vdash \Box\,\mathcal{V}(a) \end{aligned}$$

Laws for the basic update modality.

$$\text{UPD-MONO}\ \frac{P \vdash Q}{\Rightarrow\!\!\!\!\mid P \vdash \Rightarrow\!\!\!\!\mid Q} \qquad \text{UPD-INTRO}\ P \vdash \Rightarrow\!\!\!\!\mid P \qquad \text{UPD-TRANS}\ \Rightarrow\!\!\!\!\mid \Rightarrow\!\!\!\!\mid P \vdash \Rightarrow\!\!\!\!\mid P \qquad \text{UPD-FRAME}\ Q * \Rightarrow\!\!\!\!\mid P \vdash \Rightarrow\!\!\!\!\mid (Q * P)$$

$$\text{UPD-UPDATE}\ \frac{a \rightsquigarrow B}{\mathsf{Own}\,(a) \vdash \Rightarrow\!\!\!\!\mid \exists b \in B.\, \mathsf{Own}\,(b)}$$

The premise in UPD-UPDATE is a *meta-level* side-condition that has to be proven about a and B.

4.4 Consistency

The consistency statement of the logic reads as follows: For any n, we have

$$\neg(\mathsf{True} \vdash (\Rightarrow\!\!\!\!\mid \triangleright)^n\, \mathsf{False})$$

where $(\Rightarrow\!\!\!\!\mid \triangleright)^n$ is short for $\Rightarrow\!\!\!\!\mid \triangleright$ being nested n times.

The reason we want a stronger consistency than the usual $\neg(\mathsf{True} \vdash \mathsf{False})$ is our modalities: it should be impossible to derive a contradiction below the modalities. For $\Box$, this follows from the elimination rule, but the other two modalities do not have an elimination rule. Hence we declare that it is impossible to derive a contradiction below any combination of these two modalities.

5 Model and semantics

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

Semantic domains. The semantic domains are interpreted as follows:

$$\begin{array}{ll@{\qquad}ll} \llbracket \mathsf{iProp} \rrbracket & \triangleq \mathit{UPred}(M) & \llbracket 1 \rrbracket & \triangleq \Delta\{()\} \\ \llbracket \mathsf{M} \rrbracket & \triangleq M & \llbracket \tau \times \tau' \rrbracket & \triangleq \llbracket \tau \rrbracket \times \llbracket \tau \rrbracket \\ & & \llbracket \tau \to \tau' \rrbracket & \triangleq \llbracket \tau \rrbracket \xrightarrow{\text{ne}} \llbracket \tau \rrbracket \end{array}$$

For the remaining base types τ defined by the signature $\mathcal{S}$, we pick an object X_τ in $\mathcal{OFE}$ and define

$$\llbracket \tau \rrbracket \triangleq X_\tau$$

For each function symbol $F : \tau_1, \ldots, \tau_n \to \tau_{n+1} \in \mathcal{F}$, we pick a function $\llbracket F \rrbracket : \llbracket \tau_1 \rrbracket \times \cdots \times \llbracket \tau_n \rrbracket \xrightarrow{\text{ne}} \llbracket \tau_{n+1} \rrbracket$.

Interpretation of assertions. $\boxed{\llbracket \Gamma \vdash t : \mathsf{iProp} \rrbracket : \llbracket \Gamma \rrbracket \xrightarrow{\text{ne}} \mathit{UPred}(M)}$

Remember that $\mathit{UPred}(M)$ is isomorphic to $M \xrightarrow{\text{mon}} \mathit{SProp}$. We are thus going to define the assertions as mapping CMRA elements to sets of step-indices.

$$\begin{aligned}
\llbracket \Gamma \vdash t =_\tau u : \mathsf{iProp} \rrbracket_\gamma &\triangleq \lambda_. \left\{ n \,\middle|\, \llbracket \Gamma \vdash t : \tau \rrbracket_\gamma \overset{n}{=} \llbracket \Gamma \vdash u : \tau \rrbracket_\gamma \right\} \\
\llbracket \Gamma \vdash \mathsf{False} : \mathsf{iProp} \rrbracket_\gamma &\triangleq \lambda_. \emptyset \\
\llbracket \Gamma \vdash \mathsf{True} : \mathsf{iProp} \rrbracket_\gamma &\triangleq \lambda_. \mathbb{N} \\
\llbracket \Gamma \vdash P \wedge Q : \mathsf{iProp} \rrbracket_\gamma &\triangleq \lambda a. \llbracket \Gamma \vdash P : \mathsf{iProp} \rrbracket_\gamma(a) \cap \llbracket \Gamma \vdash Q : \mathsf{iProp} \rrbracket_\gamma(a) \\
\llbracket \Gamma \vdash P \vee Q : \mathsf{iProp} \rrbracket_\gamma &\triangleq \lambda a. \llbracket \Gamma \vdash P : \mathsf{iProp} \rrbracket_\gamma(a) \cup \llbracket \Gamma \vdash Q : \mathsf{iProp} \rrbracket_\gamma(a) \\
\llbracket \Gamma \vdash P \Rightarrow Q : \mathsf{iProp} \rrbracket_\gamma &\triangleq \lambda a. \left\{ n \,\middle|\, \begin{array}{r} \forall m, b.\, m \le n \wedge a \preccurlyeq b \wedge b \in \mathcal{V}_m \Rightarrow \\ m \in \llbracket \Gamma \vdash P : \mathsf{iProp} \rrbracket_\gamma(b) \Rightarrow \\ m \in \llbracket \Gamma \vdash Q : \mathsf{iProp} \rrbracket_\gamma(b) \end{array} \right\} \\
\llbracket \Gamma \vdash \forall x : \tau.\, P : \mathsf{iProp} \rrbracket_\gamma &\triangleq \lambda a. \left\{ n \,\middle|\, \forall v \in \llbracket \tau \rrbracket.\, n \in \llbracket \Gamma, x : \tau \vdash P : \mathsf{iProp} \rrbracket_{\gamma[x \leftarrow v]}(a) \right\} \\
\llbracket \Gamma \vdash \exists x : \tau.\, P : \mathsf{iProp} \rrbracket_\gamma &\triangleq \lambda a. \left\{ n \,\middle|\, \exists v \in \llbracket \tau \rrbracket.\, n \in \llbracket \Gamma, x : \tau \vdash P : \mathsf{iProp} \rrbracket_{\gamma[x \leftarrow v]}(a) \right\}
\end{aligned}$$

$$\llbracket \Gamma \vdash P * Q : \mathsf{iProp} \rrbracket_\gamma \triangleq \lambda a.\ \left\{ n \;\middle|\; \begin{array}{l} \exists b_1, b_2.\, a \overset{n}{=} b_1 \cdot b_2 \wedge \\ \qquad n \in \llbracket \Gamma \vdash P : \mathsf{iProp} \rrbracket_\gamma(b_1) \wedge n \in \llbracket \Gamma \vdash Q : \mathsf{iProp} \rrbracket_\gamma(b_2) \end{array} \right\}$$

$$\llbracket \Gamma \vdash P \mathrel{-\!\!*} Q : \mathsf{iProp} \rrbracket_\gamma \triangleq \lambda a.\ \left\{ n \;\middle|\; \begin{array}{l} \forall m, b.\, m \leq n \wedge a \cdot b \in \mathcal{V}_m \Rightarrow \\ \qquad m \in \llbracket \Gamma \vdash P : \mathsf{iProp} \rrbracket_\gamma(b) \Rightarrow \\ \qquad m \in \llbracket \Gamma \vdash Q : \mathsf{iProp} \rrbracket_\gamma(a \cdot b) \end{array} \right\}$$

$$\llbracket \Gamma \vdash \Box P : \mathsf{iProp} \rrbracket_\gamma \triangleq \lambda a.\ \llbracket \Gamma \vdash P : \mathsf{iProp} \rrbracket_\gamma(|a|)$$

$$\llbracket \Gamma \vdash \triangleright P : \mathsf{iProp} \rrbracket_\gamma \triangleq \lambda a.\ \{n \mid n = 0 \vee n - 1 \in \llbracket \Gamma \vdash P : \mathsf{iProp} \rrbracket_\gamma(a)\}$$

$$\llbracket \Gamma \vdash \mathsf{Own}\,(t) : \mathsf{iProp} \rrbracket_\gamma \triangleq \lambda b.\ \left\{ n \;\middle|\; \llbracket \Gamma \vdash t : \mathsf{M} \rrbracket_\gamma \overset{n}{\preccurlyeq} b \right\}$$

$$\llbracket \Gamma \vdash \mathcal{V}(t) : \mathsf{iProp} \rrbracket_\gamma \triangleq \lambda\_.\ \{n \mid \llbracket \Gamma \vdash t : \mathsf{M} \rrbracket_\gamma \in \mathcal{V}_n\}$$

$$\llbracket \Gamma \vdash \Rrightarrow P : \mathsf{iProp} \rrbracket_\gamma \triangleq \lambda a.\ \left\{ n \;\middle|\; \begin{array}{l} \forall m, a'.\, m \leq n \wedge (a \cdot a') \in \mathcal{V}_m \Rightarrow \\ \qquad \exists b.\, (b \cdot a') \in \mathcal{V}_m \wedge m \in \llbracket \Gamma \vdash P : \mathsf{iProp} \rrbracket_\gamma(b) \end{array} \right\}$$

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

Interpretation of non-propositional terms $\boxed{\llbracket \Gamma \vdash t : \tau \rrbracket : \llbracket \Gamma \rrbracket \xrightarrow{\text{ne}} \llbracket \tau \rrbracket}$

$$\begin{aligned}
\llbracket \Gamma \vdash x : \tau \rrbracket_\gamma &\triangleq \gamma(x) \\
\llbracket \Gamma \vdash F(t_1, \ldots, t_n) : \tau_{n+1} \rrbracket_\gamma &\triangleq \llbracket F \rrbracket(\llbracket \Gamma \vdash t_1 : \tau_1 \rrbracket_\gamma, \ldots, \llbracket \Gamma \vdash t_n : \tau_n \rrbracket_\gamma) \\
\llbracket \Gamma \vdash \lambda x : \tau.\, t : \tau \to \tau' \rrbracket_\gamma &\triangleq \lambda u : \llbracket \tau \rrbracket.\, \llbracket \Gamma, x : \tau \vdash t : \tau \rrbracket_{\gamma[x \leftarrow u]} \\
\llbracket \Gamma \vdash t(u) : \tau' \rrbracket_\gamma &\triangleq \llbracket \Gamma \vdash t : \tau \to \tau' \rrbracket_\gamma(\llbracket \Gamma \vdash u : \tau \rrbracket_\gamma) \\
\llbracket \Gamma \vdash \mu\, x : \tau.\, t : \tau \rrbracket_\gamma &\triangleq \mathit{fix}(\lambda u : \llbracket \tau \rrbracket.\, \llbracket \Gamma, x : \tau \vdash t : \tau \rrbracket_{\gamma[x \leftarrow u]})
\end{aligned}$$

$$\begin{aligned}
\llbracket \Gamma \vdash () : 1 \rrbracket_\gamma &\triangleq () \\
\llbracket \Gamma \vdash (t_1, t_2) : \tau_1 \times \tau_2 \rrbracket_\gamma &\triangleq (\llbracket \Gamma \vdash t_1 : \tau_1 \rrbracket_\gamma, \llbracket \Gamma \vdash t_2 : \tau_2 \rrbracket_\gamma) \\
\llbracket \Gamma \vdash \pi_i(t) : \tau_i \rrbracket_\gamma &\triangleq \pi_i(\llbracket \Gamma \vdash t : \tau_1 \times \tau_2 \rrbracket_\gamma)
\end{aligned}$$

$$\begin{aligned}
\llbracket a : \mathsf{M} \rrbracket_\gamma &\triangleq a \\
\llbracket \Gamma \vdash |t| : \mathsf{M} \rrbracket_\gamma &\triangleq |\llbracket \Gamma \vdash t : \mathsf{M} \rrbracket_\gamma| \\
\llbracket \Gamma \vdash t \cdot u : \mathsf{M} \rrbracket_\gamma &\triangleq \llbracket \Gamma \vdash t : \mathsf{M} \rrbracket_\gamma \cdot \llbracket \Gamma \vdash u : \mathsf{M} \rrbracket_\gamma
\end{aligned}$$

An environment Γ is interpreted as the set of finite partial functions ρ, with $\mathrm{dom}(\rho) = \mathrm{dom}(\Gamma)$ and $\rho(x) \in \llbracket \Gamma(x) \rrbracket$.

Logical entailment. We can now define *semantic* logical entailment.

Interpretation of entailment $\boxed{[\![\Gamma \mid \Theta \vdash P]\!] : \mathit{Prop}}$

$$[\![\Gamma \mid P \vdash Q]\!] \triangleq \forall n \in \mathbb{N}.\, \forall r \in M.\, \forall \gamma \in [\![\Gamma]\!],\\ n \in [\![\Gamma \vdash P : \mathsf{iProp}]\!]_\gamma(r) \Rightarrow n \in [\![\Gamma \vdash Q : \mathsf{iProp}]\!]_\gamma(r)$$

The following soundness theorem connects syntactic and semantic entailment. It is proven by showing that all the syntactic proof rules of §4 can be validated in the model.

$$\Gamma \mid P \vdash Q \Rightarrow [\![\Gamma \mid P \vdash Q]\!]$$

It now becomes straight-forward to show consistency of the logic.

6 Extensions of the Base Logic

In this section we discuss some additional constructions that we define within and on top of the base logic. These are not "extensions" in the sense that they change the proof power of the logic, they just form useful derived principles.

6.1 Derived rules about base connectives

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

$$P \Rightarrow Q \vdash P \mathrel{-\!\!*} Q \qquad P * \exists x.\, Q \dashv\vdash \exists x.\, P * Q \qquad P * \forall x.\, Q \vdash \forall x.\, P * Q \qquad \Box(P * Q) \dashv\vdash \Box P * \Box Q$$

$$\Box(P \Rightarrow Q) \vdash \Box P \Rightarrow \Box Q \qquad \Box(P \mathrel{-\!\!*} Q) \vdash \Box P \mathrel{-\!\!*} \Box Q \qquad \Box(P \mathrel{-\!\!*} Q) \dashv\vdash \Box(P \Rightarrow Q)$$

$$\triangleright(P \Rightarrow Q) \vdash \triangleright P \Rightarrow \triangleright Q \qquad \triangleright(P \mathrel{-\!\!*} Q) \vdash \triangleright P \mathrel{-\!\!*} \triangleright Q \qquad P \vdash \triangleright P$$

6.2 Persistent assertions

We call an assertion P *persistent* if $P \vdash \Box P$. These are assertions that "don't own anything", so we can (and will) treat them like "normal" intuitionistic assertions.

Of course, $\Box P$ is persistent for any P. Furthermore, by the proof rules given in §4.3, True, False, $t = t'$ as well as $\boxed{\boxed{a}}^{\gamma}$ and $\mathcal{V}(a)$ are persistent. Persistence is preserved by conjunction, disjunction, separating conjunction as well as universal and existential quantification and $\triangleright$.

6.3 Timeless assertions and except-0

One of the troubles of working in a step-indexed logic is the "later" modality $\triangleright$. It turns out that we can somewhat mitigate this trouble by working below the following *except-0* modality:

$$\diamond P \triangleq \triangleright \mathsf{False} \vee P$$

This modality is useful because there is a class of assertions which we call *timeless* assertions, for which we have

$$\mathsf{timeless}(P) \triangleq \triangleright P \vdash \diamond P$$

In other words, when working below the except-0 modality, we can *strip away* the later from timeless assertions.

The following rules can be derived about except-0:

$$\textsc{ex0-mono}\ \frac{P \vdash Q}{\diamond P \vdash \diamond Q} \qquad \textsc{ex0-intro}\ P \vdash \diamond P \qquad \textsc{ex0-idem}\ \diamond\diamond P \vdash \diamond P$$

$$\diamond(P * Q) \dashv\vdash \diamond P * \diamond Q \qquad \diamond(P \wedge Q) \dashv\vdash \diamond P \wedge \diamond Q \qquad \diamond(P \vee Q) \dashv\vdash \diamond P \vee \diamond Q$$

$$\diamond\forall x.\, P \dashv\vdash \forall x.\, \diamond P \qquad \diamond\exists x.\, P \dashv\vdash \exists x.\, \diamond P \qquad \diamond\Box P \dashv\vdash \Box\diamond P \qquad \diamond\triangleright P \vdash \triangleright P$$

The following rules identify the class of timeless assertions:

$$\frac{\Gamma \vdash \mathsf{timeless}(P) \qquad \Gamma \vdash \mathsf{timeless}(Q)}{\Gamma \vdash \mathsf{timeless}(P \wedge Q)} \qquad \frac{\Gamma \vdash \mathsf{timeless}(P) \qquad \Gamma \vdash \mathsf{timeless}(Q)}{\Gamma \vdash \mathsf{timeless}(P \vee Q)}$$

$$\frac{\Gamma \vdash \mathsf{timeless}(P) \qquad \Gamma \vdash \mathsf{timeless}(Q)}{\Gamma \vdash \mathsf{timeless}(P * Q)} \qquad \frac{\Gamma \vdash \mathsf{timeless}(P)}{\Gamma \vdash \mathsf{timeless}(\Box P)} \qquad \frac{\Gamma \vdash \mathsf{timeless}(Q)}{\Gamma \vdash \mathsf{timeless}(P \Rightarrow Q)}$$

$$\frac{\Gamma \vdash \mathsf{timeless}(Q)}{\Gamma \vdash \mathsf{timeless}(P \mathrel{-\!\!*} Q)} \qquad \frac{\Gamma, x : \tau \vdash \mathsf{timeless}(P)}{\Gamma \vdash \mathsf{timeless}(\forall x : \tau.\, P)} \qquad \frac{\Gamma, x : \tau \vdash \mathsf{timeless}(P)}{\Gamma \vdash \mathsf{timeless}(\exists x : \tau.\, P)} \qquad \mathsf{timeless}(\mathsf{True})$$

$$\mathsf{timeless}(\mathsf{False}) \qquad \frac{t \text{ or } t' \text{ is a discrete OFE element}}{\mathsf{timeless}(t =_\tau t')} \qquad \frac{a \text{ is a discrete OFE element}}{\mathsf{timeless}(\mathsf{Own}\,(a))}$$

$$\frac{a \text{ is an element of a discrete CMRA}}{\mathsf{timeless}(\mathcal{V}(a))}$$

7 Language

A *language* Λ consists of a set *Expr* of *expressions* (metavariable e), a set *Val* of *values* (metavariable v), and a nonempty set *State* of *states* (metavariable σ) such that

- There exist functions val_to_expr : $\mathit{Val} \to \mathit{Expr}$ and expr_to_val : $\mathit{Expr} \rightharpoonup \mathit{Val}$ (notice the latter is partial), such that

  $$\forall e, v.\, \text{expr\_to\_val}(e) = v \Rightarrow \text{val\_to\_expr}(v) = e \qquad \forall v.\, \text{expr\_to\_val}(\text{val\_to\_expr}(v)) = v$$

- There exists a *primitive reduction relation*

  $$(-, - \to_{\mathsf{t}} -, -, -) \subseteq \mathit{Expr} \times \mathit{State} \times \mathit{Expr} \times \mathit{State} \times \mathit{List}(\mathit{Expr})$$

  A reduction $e_1, \sigma_1 \to_{\mathsf{t}} e_2, \sigma_2, \overline{e}$ indicates that, when e_1 reduces to e_2, the new threads in the list $\overline{e}$ is forked off. We will write $e_1, \sigma_1 \to_{\mathsf{t}} e_2, \sigma_2$ for $e_1, \sigma_1 \to_{\mathsf{t}} e_2, \sigma_2, ()$, *i.e.,* when no threads are forked off.

- All values are stuck:

  $$e, \_ \to_{\mathsf{t}} \_, \_, \_ \Rightarrow \text{expr\_to\_val}(e) = \bot$$

Definition 19. *An expression e and state σ are* reducible *(written* $\text{red}(e, \sigma)$*) if*

$$\exists e_2, \sigma_2, \vec{e}.\, e, \sigma \to_{\mathsf{t}} e_2, \sigma_2, \vec{e}$$

Definition 20. *An expression e is* atomic *if it reduces in one step to something irreducible:*

$$\forall \sigma_1, e_2, \sigma_2, \vec{e}.\, e, \sigma_1 \to_{\mathsf{t}} e_2, \sigma_2, \vec{e} \Rightarrow \neg\text{red}(e_2, \sigma_2)$$

Definition 21 (Context)**.** *A function $K : \mathit{Expr} \to \mathit{Expr}$ is a* context *if the following conditions are satisfied:*

1. *K does not turn non-values into values:*
   $\forall e.\, \text{expr\_to\_val}(e) = \bot \Rightarrow \text{expr\_to\_val}(K(e)) = \bot$
2. *One can perform reductions below K:*
   $\forall e_1, \sigma_1, e_2, \sigma_2, \vec{e}.\, e_1, \sigma_1 \to_{\mathsf{t}} e_2, \sigma_2, \vec{e} \Rightarrow K(e_1), \sigma_1 \to_{\mathsf{t}} K(e_2), \sigma_2, \vec{e}$
3. *Reductions stay below K until there is a value in the hole:*
   $\forall e_1', \sigma_1, e_2, \sigma_2, \vec{e}.\, \text{expr\_to\_val}(e_1') = \bot \wedge K(e_1'), \sigma_1 \to_{\mathsf{t}} e_2, \sigma_2, \vec{e} \Rightarrow \exists e_2'.\, e_2 = K(e_2') \wedge e_1', \sigma_1 \to_{\mathsf{t}} e_2', \sigma_2, \vec{e}$

7.1 Concurrent language

For any language Λ, we define the corresponding thread-pool semantics.

Machine syntax

$$T \in \mathit{ThreadPool} \triangleq \mathit{List}(\mathit{Expr})$$

Machine reduction $\boxed{T; \sigma \to_{\mathsf{t}} T'; \sigma'}$

$$\frac{e_1, \sigma_1 \to_{\mathsf{t}} e_2, \sigma_2, \vec{e}}{T +\!\!+ [e_1] +\!\!+ T'; \sigma_1 \to_{\mathsf{t}} T +\!\!+ [e_2] +\!\!+ T' +\!\!+ \vec{e}; \sigma_2}$$

8 Program Logic

This section describes how to build a program logic for an arbitrary language (*c.f.* §7) on top of the base logic. So in the following, we assume that some language Λ was fixed.

8.1 Dynamic Composeable Higher-Order Resources

The base logic described in §4 works over an arbitrary CMRA M defining the structure of the resources. It turns out that we can generalize this further and permit picking CMRAs "$\Sigma(\mathsf{iProp})$" that depend on the structure of assertions themselves. Of course, iProp is just the syntactic type of assertions; for this to make sense we have to look at the semantics.

Furthermore, there is a composability problem with the given logic: if we have one proof performed with CMRA M_1, and another proof carried out with a *different* CMRA M_2, then the two proofs are actually carried out in two *entirely separate logics* and hence cannot be combined.

Finally, in many cases just having a single "instance" of a CMRA available for reasoning is not enough. For example, when reasoning about a dynamically allocated data structure, every time a new instance of that data structure is created, we will want a fresh resource governing the state of this particular instance. While it would be possible to handle this problem whenever it comes up, it turns out to be useful to provide a general solution.

The purpose of this section is to describe how we solve these issues.

Picking the resources. The key ingredient that we will employ on top of the base logic is to give some more fixed structure to the resources. To instantiate the program logic, the user picks a family of locally contractive bifunctors $(\Sigma_i : \mathcal{OFE} \to \mathcal{CMRA})_{i \in \mathcal{I}}$. (This is in contrast to the base logic, where the user picks a single, fixed CMRA that has a unit.)

From this, we construct the bifunctor defining the overall resources as follows:

$$\begin{aligned} \mathit{GName} &\triangleq \mathbb{N} \\ \mathit{ResF}(T^{\mathrm{op}}, T) &\triangleq \prod_{i \in \mathcal{I}} \mathit{GName} \xrightarrow{\mathrm{fin}} \Sigma_i(T^{\mathrm{op}}, T) \end{aligned}$$

We will motivate both the use of a product and the finite partial function below. $\mathit{ResF}(T^{\mathrm{op}}, T)$ is a CMRA by lifting the individual CMRAs pointwise, and it has a unit (using the empty finite partial functions). Furthermore, since the Σ_i are locally contractive, so is ResF.

Now we can write down the recursive domain equation:

$$\mathit{iPreProp} \cong \mathit{UPred}(\mathit{ResF}(\mathit{iPreProp}, \mathit{iPreProp}))$$

iPreProp is a COFE defined as the fixed-point of a locally contractive bifunctor. This fixed-point exists and is unique[2] 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

$$\begin{aligned} \mathit{Res} &\triangleq \mathit{ResF}(\mathit{iPreProp}, \mathit{iPreProp}) \\ \mathit{iProp} &\triangleq \mathit{UPred}(\mathit{Res}) \\ \xi &: \mathit{iProp} \xrightarrow{\mathrm{ne}} \mathit{iPreProp} \\ \xi^{-1} &: \mathit{iPreProp} \xrightarrow{\mathrm{ne}} \mathit{iProp} \end{aligned}$$

[2] We have not proven uniqueness in Coq.

Notice that *iProp* is the semantic model of assertions for the base logic described in §4 with *Res*:

$$[\![\mathsf{iProp}]\!] \triangleq \mathit{iProp} = \mathit{UPred}(\mathit{Res})$$

Effectively, we just defined a way to instantiate the base logic with *Res* as the CMRA of resources, while providing a way for *Res* to depend on *iPreProp*, which is isomorphic to $[\![\mathsf{iProp}]\!]$.

We thus obtain all the rules of §4, and furthermore, we can use the maps ξ and ξ^{-1} *in the logic* to convert between logical assertions $[\![\mathsf{iProp}]\!]$ and the domain *iPreProp* which is used in the construction of *Res* – so from elements of *iPreProp*, we can construct elements of $[\![\mathsf{M}]\!]$, which are the elements that can be owned in our logic.

Proof composability. To make our proofs composeable, we *generalize* our proofs over the family of functors. This is possible because we made *Res* a *product* of all the CMRAs picked by the user, and because we can actually work with that product "pointwise". So instead of picking a *concrete* family, proofs will assume to be given an *arbitrary* family of functors, plus a proof that this family *contains the functors they need*. Composing two proofs is then merely a matter of conjoining the assumptions they make about the functors. Since the logic is entirely parametric in the choice of functors, there is no trouble reasoning without full knowledge of the family of functors.

Only when the top-level proof is completed we will "close" the proof by picking a concrete family that contains exactly those functors the proof needs.

Dynamic resources. Finally, the use of finite partial functions lets us have as many instances of any CMRA as we could wish for: Because there can only ever be finitely many instances already allocated, it is always possible to create a fresh instance with any desired (valid) starting state. This is best demonstrated by giving some proof rules.

So let us first define the notion of ghost ownership that we use in this logic. Assuming that the family of functors contains the functor Σ_i at index i, and furthermore assuming that $M_i = \Sigma_i(\mathit{iPreProp}, \mathit{iPreProp})$, given some $a \in M_i$ we define:

$$\boxed{a : M_i}^{\gamma} \triangleq \mathsf{Own}\left(\left(\ldots, \emptyset, i : [\gamma \leftarrow a], \emptyset, \ldots\right)\right)$$

This is ownership of the pair (element of the product over all the functors) that has the empty finite partial function in all components *except for* the component corresponding to index i, where we own the element a at index γ in the finite partial function.

We can show the following properties for this form of ownership:

RES-ALLOC
$$\frac{G \text{ infinite} \qquad a \in \mathcal{V}_{M_i}}{\mathsf{True} \vdash \Rrightarrow \exists \gamma \in G.\, \boxed{a : M_i}^{\gamma}}$$

RES-UPDATE
$$\frac{a \rightsquigarrow_{M_i} B}{\boxed{a : M_i}^{\gamma} \vdash \Rrightarrow \exists b \in B.\, \boxed{b : M_i}^{\gamma}}$$

RES-EMPTY
$$\frac{\varepsilon \text{ is a unit of } M_i}{\mathsf{True} \vdash \Rrightarrow \boxed{\varepsilon}^{\gamma}}$$

RES-OP
$$\boxed{a : M_i}^{\gamma} * \boxed{b : M_i}^{\gamma} \dashv\vdash \boxed{a \cdot b : M_i}^{\gamma}$$

RES-VALID
$$\boxed{a : M_i}^{\gamma} \Rightarrow \mathcal{V}_{M_i}(a)$$

RES-TIMELESS
$$\frac{a \text{ is a discrete OFE element}}{\mathsf{timeless}(\boxed{a : M_i}^{\gamma})}$$

Below, we will always work within (an instance of) the logic as described here. Whenever a CMRA is used in a proof, we implicitly assume it to be available in the global family of functors. We will typically leave the M_i implicit when asserting ghost ownership, as the type of a will be clear from the context.

8.2 World Satisfaction, Invariants, Fancy Updates

To introduce invariants into our logic, we will define weakest precondition to explicitly thread through the proof that all the invariants are maintained throughout program execution. However, in order to be able to access invariants, we will also have to provide a way to *temporarily disable* (or "open") them. To this end, we use tokens that manage which invariants are currently enabled.

We assume to have the following four CMRAs available:

$$\begin{aligned} \mathit{InvName} &\triangleq \mathbb{N} \\ \textsc{Inv} &\triangleq \textsc{Auth}(\mathit{InvName} \xrightarrow{\text{fin}} \textsc{Ag}(\blacktriangleright \mathit{iPreProp})) \\ \textsc{En} &\triangleq \wp(\mathit{InvName}) \\ \textsc{Dis} &\triangleq \wp^{\text{fin}}(\mathit{InvName}) \end{aligned}$$

The last two are the tokens used for managing invariants, Inv is the monoid used to manage the invariants themselves.

We assume that at the beginning of the verification, instances named $\gamma_{\textsc{State}}$, $\gamma_{\textsc{Inv}}$, $\gamma_{\textsc{En}}$ and $\gamma_{\textsc{Dis}}$ of these CMRAs have been created, such that these names are globally known.

World Satisfaction. We can now define the assertion W (*world satisfaction*) which ensures that the enabled invariants are actually maintained:

$$W \triangleq \exists I : \mathit{InvName} \xrightarrow{\text{fin}} \mathsf{iProp}.\ \boxed{\bullet \left[\iota \leftarrow \mathsf{ag}(\mathsf{next}(\xi(I(\iota)))) \mid \iota \in \mathrm{dom}(I)\right]}^{\gamma_{\textsc{Inv}}} * \ast_{\iota \in \mathrm{dom}(I)} \left(\triangleright I(\iota) * \boxed{\{\iota\}}^{\gamma_{\textsc{Dis}}} \vee \boxed{\{\iota\}}^{\gamma_{\textsc{En}}}\right)$$

Invariants. The following assertion states that an invariant with name ι exists and maintains assertion P:

$$\boxed{P}^{\iota} \triangleq \boxed{\circ \left[\iota \leftarrow \mathsf{ag}(\mathsf{next}(\xi(P)))\right]}^{\gamma_{\textsc{Inv}}}$$

Fancy Updates and View Shifts. Next, we define *fancy updates*, which are essentially the same as the basic updates of the base logic (§4), except that they also have access to world satisfaction and can enable and disable invariants:

$$\prescript{\mathcal{E}_1}{}{\Rightarrow}^{\mathcal{E}_2} P \triangleq W * \boxed{\mathcal{E}_1}^{\gamma_{\textsc{En}}} \mathrel{-\!\!*} \Rightarrow \diamond (W * \boxed{\mathcal{E}_2}^{\gamma_{\textsc{En}}} * P)$$

Here, $\mathcal{E}_1$ and $\mathcal{E}_2$ are the *masks* of the view update, defining which invariants have to be (at least!) available before and after the update. We use $\top$ as symbol for the largest possible mask, $\mathbb{N}$, and $\bot$ for the smallest possible mask $\emptyset$. We will write $\Rightarrow_{\mathcal{E}} P$ for $\prescript{\mathcal{E}}{}{\Rightarrow}^{\mathcal{E}} P$. Fancy updates satisfy the following basic proof rules:

FUP-MONO
$$\frac{P \vdash Q}{\prescript{\mathcal{E}_1}{}{\Rightarrow}^{\mathcal{E}_2} P \vdash \prescript{\mathcal{E}_1}{}{\Rightarrow}^{\mathcal{E}_2} Q}$$

FUP-INTRO-MASK
$$\frac{\mathcal{E}_2 \subseteq \mathcal{E}_1}{P \vdash \prescript{\mathcal{E}_1}{}{\Rightarrow}^{\mathcal{E}_2} \prescript{\mathcal{E}_2}{}{\Rightarrow}^{\mathcal{E}_1} P}$$

FUP-TRANS
$$\prescript{\mathcal{E}_1}{}{\Rightarrow}^{\mathcal{E}_2} \prescript{\mathcal{E}_2}{}{\Rightarrow}^{\mathcal{E}_3} P \vdash \prescript{\mathcal{E}_1}{}{\Rightarrow}^{\mathcal{E}_3} P$$

FUP-UPD
$$\Rightarrow P \vdash \Rightarrow_{\mathcal{E}} P$$

FUP-FRAME
$$Q * \prescript{\mathcal{E}_1}{}{\Rightarrow}^{\mathcal{E}_2} P \vdash \prescript{\mathcal{E}_1 \uplus \mathcal{E}_{\mathrm{f}}}{}{\Rightarrow}^{\mathcal{E}_2 \uplus \mathcal{E}_{\mathrm{f}}} Q * P$$

FUP-UPDATE
$$\frac{a \rightsquigarrow B}{\mathsf{Own}\,(a) \vdash \Rightarrow_{\mathcal{E}} \exists b \in B.\, \mathsf{Own}\,(b)}$$

FUP-TIMELESS
$$\frac{\mathsf{timeless}(P)}{\triangleright P \vdash \Rightarrow_{\mathcal{E}} P}$$

(There are no rules related to invariants here. Those rules will be discussed later, in §8.4.)

We can further define the notions of *view shifts* and *linear view shifts*:

$$P \ {}^{\mathcal{E}_1}\!\Rrightarrow\!\!\!\!\!\!\ast^{\mathcal{E}_2} \ Q \triangleq P \mathrel{-\!\!*} {}^{\mathcal{E}_1}\!\Rrightarrow^{\mathcal{E}_2} Q$$
$$P \ {}^{\mathcal{E}_1}\!\Rrightarrow^{\mathcal{E}_2} \ Q \triangleq \Box(P \mathrel{-\!\!*} {}^{\mathcal{E}_1}\!\Rrightarrow^{\mathcal{E}_2} Q)$$
$$P \Rrightarrow_{\mathcal{E}} Q \triangleq P \ {}^{\mathcal{E}}\!\Rrightarrow^{\mathcal{E}} \ Q$$

These two are useful when writing down specifications and for comparing with previous versions of Iris, but for reasoning, it is typically easier to just work directly with fancy updates. Still, just to give an idea of what view shifts "are", here are some proof rules for them:

VS-UPDATE
$$\frac{a \rightsquigarrow B}{\boxed{a}^{\gamma} \Rrightarrow_{\emptyset} \exists b \in B.\, \boxed{b}^{\gamma}}$$

VS-TRANS
$$\frac{P \ {}^{\mathcal{E}_1}\!\Rrightarrow^{\mathcal{E}_2} \ Q \qquad Q \ {}^{\mathcal{E}_2}\!\Rrightarrow^{\mathcal{E}_3} \ R}{P \ {}^{\mathcal{E}_1}\!\Rrightarrow^{\mathcal{E}_3} \ R}$$

VS-IMP
$$\frac{\Box(P \Rightarrow Q)}{P \Rrightarrow_{\emptyset} Q}$$

VS-MASK-FRAME
$$\frac{P \ {}^{\mathcal{E}_1}\!\Rrightarrow^{\mathcal{E}_2} \ Q}{P \ {}^{\mathcal{E}_1 \uplus \mathcal{E}'}\!\Rrightarrow^{\mathcal{E}_2 \uplus \mathcal{E}'} \ Q}$$

VS-FRAME
$$\frac{P \ {}^{\mathcal{E}_1}\!\Rrightarrow^{\mathcal{E}_2} \ Q}{P * R \ {}^{\mathcal{E}_1}\!\Rrightarrow^{\mathcal{E}_2} \ Q * R}$$

VS-TIMELESS
$$\frac{\mathsf{timeless}(P)}{\triangleright P \Rrightarrow_{\emptyset} P}$$

VS-DISJ
$$\frac{P \ {}^{\mathcal{E}_1}\!\Rrightarrow^{\mathcal{E}_2} \ R \qquad Q \ {}^{\mathcal{E}_1}\!\Rrightarrow^{\mathcal{E}_2} \ R}{P \vee Q \ {}^{\mathcal{E}_1}\!\Rrightarrow^{\mathcal{E}_2} \ R}$$

VS-EXIST
$$\frac{\forall x.\, (P \ {}^{\mathcal{E}_1}\!\Rrightarrow^{\mathcal{E}_2} \ Q)}{(\exists x.\, P) \ {}^{\mathcal{E}_1}\!\Rrightarrow^{\mathcal{E}_2} \ Q}$$

VS-ALWAYS
$$\frac{\Box Q \vdash P \ {}^{\mathcal{E}_1}\!\Rrightarrow^{\mathcal{E}_2} \ R}{P \wedge \Box Q \ {}^{\mathcal{E}_1}\!\Rrightarrow^{\mathcal{E}_2} \ R}$$

VS-FALSE
$$\mathsf{False} \ {}^{\mathcal{E}_1}\!\Rrightarrow^{\mathcal{E}_2} \ P$$

8.3 Weakest Precondition

Finally, we can define the core piece of the program logic, the assertion that reasons about program behavior: Weakest precondition, from which Hoare triples will be derived.

Defining weakest precondition. We assume that everything making up the definition of the language, *i.e.,* values, expressions, states, the conversion functions, reduction relation and all their properties, are suitably reflected into the logic (*i.e.,* they are part of the signature $\mathcal{S}$). We further assume (as a parameter) a predicate $I : \mathit{State} \to \mathit{iProp}$ that interprets the physical state as an Iris assertion. This can be instantiated, for example, with ownership of an authoritative RA to tie the physical state to fragments that are used for user-level proofs.

$$\begin{aligned}
wp \triangleq \mu\, wp.\, \lambda \mathcal{E}, e, \Phi.\ & \\
& (\exists v.\, \mathrm{expr_to_val}(e) = v \wedge \Rrightarrow_{\mathcal{E}} \Phi(v)) \vee \\
& \Big(\mathrm{expr_to_val}(e) = \bot \wedge \forall \sigma.\, I(\sigma) \ {}^{\mathcal{E}}\!\Rrightarrow\!\!\!\!\!\!\ast^{\emptyset} \\
& \qquad \mathrm{red}(e, \sigma) * \triangleright \forall e', \sigma', \vec{e}.\, (e, \sigma \to_{\mathsf{t}} e', \sigma', \vec{e}) \ {}^{\emptyset}\!\Rrightarrow\!\!\!\!\!\!\ast^{\mathcal{E}} \\
& \qquad\qquad I(\sigma') * wp(\mathcal{E}, e', \Phi) * \ast_{e'' \in \vec{e}} wp(\top, e'', \lambda_.\, \mathsf{True})\Big)
\end{aligned}$$

$$\mathsf{wp}_{\mathcal{E}}\, e\, \{v.\, P\} \triangleq wp(\mathcal{E}, e, \lambda v.\, P)$$

If we leave away the mask, we assume it to default to $\top$.

Laws of weakest precondition. The following rules can all be derived:

WP-VALUE
$$P[v/x] \vdash \mathsf{wp}_{\mathcal{E}}\, v\, \{x.\, P\}$$

WP-MONO
$$\frac{\mathcal{E}_1 \subseteq \mathcal{E}_2 \qquad \Gamma, x : \mathsf{val} \mid P \vdash Q}{\Gamma \mid \mathsf{wp}_{\mathcal{E}_1}\, e\, \{x.\, P\} \vdash \mathsf{wp}_{\mathcal{E}_2}\, e\, \{x.\, Q\}}$$

FUP-WP
$$\Rrightarrow_{\mathcal{E}} \mathsf{wp}_{\mathcal{E}}\, e\, \{x.\, P\} \vdash \mathsf{wp}_{\mathcal{E}}\, e\, \{x.\, P\}$$

WP-FUP
$$\mathsf{wp}_{\mathcal{E}}\, e\, \{x.\, \Rrightarrow_{\mathcal{E}} P\} \vdash \mathsf{wp}_{\mathcal{E}}\, e\, \{x.\, P\}$$

WP-ATOMIC
$$\frac{\mathsf{atomic(e)}}{{}^{\mathcal{E}_1}\!\Rrightarrow^{\mathcal{E}_2} \mathsf{wp}_{\mathcal{E}_2}\, e \left\{x.\, {}^{\mathcal{E}_2}\!\Rrightarrow^{\mathcal{E}_1} P\right\} \vdash \mathsf{wp}_{\mathcal{E}_1}\, e\, \{x.\, P\}}$$

WP-FRAME
$$Q * \mathsf{wp}_{\mathcal{E}}\, e\, \{x.\, P\} \vdash \mathsf{wp}_{\mathcal{E}}\, e\, \{x.\, Q * P\}$$

WP-FRAME-STEP
$$\frac{\mathrm{expr\_to\_val}(e) = \bot \qquad \mathcal{E}_2 \subseteq \mathcal{E}_1}{\mathsf{wp}_{\mathcal{E}_2}\, e\, \{x.\, P\} * {}^{\mathcal{E}_1}\!\Rrightarrow^{\mathcal{E}_2} \triangleright\, {}^{\mathcal{E}_2}\!\Rrightarrow^{\mathcal{E}_1} Q \vdash \mathsf{wp}_{\mathcal{E}_1}\, e\, \{x.\, Q * P\}}$$

WP-BIND
$$\frac{K \text{ is a context}}{\mathsf{wp}_{\mathcal{E}}\, e\, \{x.\, \mathsf{wp}_{\mathcal{E}}\, K(\mathrm{val\_to\_expr}(x))\, \{y.\, P\}\} \vdash \mathsf{wp}_{\mathcal{E}}\, K(e)\, \{y.\, P\}}$$

We will also want a rule that connect weakest preconditions to the operational semantics of the language.

WP-LIFT-STEP
$$\frac{\mathrm{expr\_to\_val}(e_1) = \bot}{\begin{array}{l} \forall \sigma_1.\, I(\sigma_1) \;{}^{\mathcal{E}}\!\Rrightarrow\!\!\!\!\!\!\ast^{\emptyset}\; \mathrm{red}(e_1, \sigma_1) * \\ \quad \triangleright \forall e_2, \sigma_2, \vec{e}.\, (e_1, \sigma_1 \to_{\mathsf{t}} e_2, \sigma_2, \vec{e})\; {}^{\emptyset}\!\Rrightarrow\!\!\!\!\!\!\ast^{\mathcal{E}} \left(I(\sigma_2) * \mathsf{wp}_{\mathcal{E}}\, e_2\, \{x.\, P\} * \ast_{e_{\mathrm{f}} \in \vec{e}} \mathsf{wp}_{\top}\, e_{\mathrm{f}}\, \{\_.\, \mathsf{True}\}\right) \\ \vdash \mathsf{wp}_{\mathcal{E}}\, e_1\, \{x.\, P\} \end{array}}$$

Adequacy of weakest precondition. The purpose of the adequacy statement is to show that our notion of weakest preconditions is *realistic* in the sense that it actually has anything to do with the actual behavior of the program. There are two properties we are looking for: First of all, the postcondition should reflect actual properties of the values the program can terminate with. Second, a proof of a weakest precondition with any postcondition should imply that the program is *safe*, *i.e.,* that it does not get stuck.

Definition 22 (Adequacy)**.** *A program e in some initial state σ is* adequate *for a set $V \subseteq Val$ of legal return values ($e, \sigma \vDash V$) if for all T', σ' such that $([e], \sigma) \to^*_{\mathsf{tp}} (T', \sigma')$ we have*

1. *Safety: For any $e' \in T'$ we have that either e' is a value, or* $\mathrm{red}(e'_i, \sigma')$*:*

$$\forall e' \in T'.\, \mathrm{expr\_to\_val}(e') \neq \bot \vee \mathrm{red}(e', \sigma')$$

   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.*

2. *Legal return value: If T'_1 (the main thread) is a value v', then $v' \in V$:*

$$\forall v', T''.\, T' = [v'] +\!\!+ T'' \Rightarrow v' \in V$$

To express the adequacy statement for functional correctness, we assume that the signature $\mathcal{S}$ adds a predicate Φ to the logic:

$$\Phi : \mathit{Val} \to \mathsf{iProp} \in \mathcal{F}$$

Furthermore, we assume that the *interpretation* $[\![\Phi]\!]$ of Φ reflects some set V of legal return values into the logic (also see §5):

$$[\![\Phi]\!] \;:\; [\![\mathit{Val}]\!] \xrightarrow{\text{ne}} [\![\mathsf{iProp}]\!]$$
$$[\![\Phi]\!] \triangleq \lambda v.\, \lambda\_.\, \{n \mid v \in V\}$$

The signature can of course state arbitrary additional properties of Φ, as long as they are proven sound. The adequacy statement now reads as follows:

$$\begin{array}{l}\forall \mathcal{E}, e, v, \sigma.\\ (\mathsf{True} \vdash \Rrightarrow_{\mathcal{E}} \exists I.\, I(\sigma) * \mathsf{wp}_{\mathcal{E}}\, e\, \{x.\, \Phi(x)\}) \Rightarrow\\ e, \sigma \vDash V\end{array}$$

Notice that the state invariant S used by the weakest precondition is chosen *after* doing a fancy update, which allows it to depend on the names of ghost variables that are picked in that initial fancy update.

Hoare triples. It turns out that weakest precondition is actually quite convenient to work with, in particular when perfoming these proofs in Coq. Still, for a more traditional presentation, we can easily derive the notion of a Hoare triple:

$$\{P\}\, e\, \{v.\, Q\}_{\mathcal{E}} \triangleq \Box\,(P \mathrel{-\!\!*} \mathsf{wp}_{\mathcal{E}}\, e\, \{v.\, Q\})$$

We only give some of the proof rules for Hoare triples here, since we usually do all our reasoning directly with weakest preconditions and use Hoare triples only to write specifications.

HT-RET
$$\{\mathsf{True}\}\, w\, \{v.\, v = w\}_{\mathcal{E}}$$

HT-BIND
$$\frac{K \text{ is a context} \qquad \{P\}\, e\, \{v.\, Q\}_{\mathcal{E}} \qquad \forall v.\, \{Q\}\, K(v)\, \{w.\, R\}_{\mathcal{E}}}{\{P\}\, K(e)\, \{w.\, R\}_{\mathcal{E}}}$$

HT-CSQ
$$\frac{P \Rrightarrow P' \qquad \{P'\}\, e\, \{v.\, Q'\}_{\mathcal{E}} \qquad \forall v.\, Q' \Rrightarrow Q}{\{P\}\, e\, \{v.\, Q\}_{\mathcal{E}}}$$

HT-FRAME
$$\frac{\{P\}\, e\, \{v.\, Q\}_{\mathcal{E}}}{\{P * R\}\, e\, \{v.\, Q * R\}_{\mathcal{E}}}$$

HT-ATOMIC
$$\frac{P \;\; {}^{\mathcal{E}\uplus\mathcal{E}'}\!\Rrightarrow^{\mathcal{E}} \; P' \qquad \{P'\}\, e\, \{v.\, Q'\}_{\mathcal{E}} \qquad \forall v.\, Q' \;\; {}^{\mathcal{E}}\!\Rrightarrow^{\mathcal{E}\uplus\mathcal{E}'} \; Q \qquad \mathsf{atomic(e)}}{\{P\}\, e\, \{v.\, Q\}_{\mathcal{E}\uplus\mathcal{E}'}}$$

HT-FALSE
$$\{\mathsf{False}\}\, e\, \{v.\, P\}_{\mathcal{E}}$$

HT-DISJ
$$\frac{\{P\}\, e\, \{v.\, R\}_{\mathcal{E}} \qquad \{Q\}\, e\, \{v.\, R\}_{\mathcal{E}}}{\{P \vee Q\}\, e\, \{v.\, R\}_{\mathcal{E}}}$$

HT-EXIST
$$\frac{\forall x.\, \{P\}\, e\, \{v.\, Q\}_{\mathcal{E}}}{\{\exists x.\, P\}\, e\, \{v.\, Q\}_{\mathcal{E}}}$$

HT-BOX
$$\frac{\Box Q \vdash \{P\}\, e\, \{v.\, R\}_{\mathcal{E}}}{\{P \wedge \Box Q\}\, e\, \{v.\, R\}_{\mathcal{E}}}$$

8.4 Invariant Namespaces

In §8.2, we defined an assertion $\boxed{P}^{\iota}$ expressing knowledge (*i.e.,* the assertion is persistent) that P is maintained as invariant with name ι. The concrete name ι is picked when the invariant is allocated, so it cannot possibly be statically known – it will always be a variable that's threaded through everything. However, we hardly care about the actual, concrete name. All we need to know is that this name is *different* from the names of other invariants that we want to open at the same time. Keeping track of the n^2 mutual inequalities that arise with n invariants quickly gets in the way of the actual proof.

To solve this issue, instead of remembering the exact name picked for an invariant, we will keep track of the *namespace* the invariant was allocated in. Namespaces are sets of invariants, following a tree-like structure: Think of the name of an invariant as a sequence of identifiers, much like a fully qualified Java class name. A *namespace* $\mathcal{N}$ then is like a Java package: it is a sequence of identifiers that we think of as *containing* all invariant names that begin with this sequence. For example, `org.mpi-sws.iris` is a namespace containing the invariant name `org.mpi-sws.iris.heap`.

The crux is that all namespaces contain infinitely many invariants, and hence we can *freely pick* the namespace an invariant is allocated in – no further, unpredictable choice has to be made. Furthermore, we will often know that namespaces are *disjoint* just by looking at them. The namespaces $\mathcal{N}$.`iris` and $\mathcal{N}$.`gps` are disjoint no matter the choice of $\mathcal{N}$. As a result, there is often no need to track disjointness of namespaces, we just have to pick the namespaces that we allocate our invariants in accordingly.

Formally speaking, let $\mathcal{N} \in \mathsf{InvNamesp} \triangleq \mathit{List}(\mathbb{N})$ be the type of *invariant namespaces*. We use the notation $\mathcal{N}.\iota$ for the namespace $[\iota] +\!\!+ \mathcal{N}$. (In other words, the list is "backwards". This is because cons-ing to the list, like the dot does above, is easier to deal with in Coq than appending at the end.)

The elements of a namespaces are *structured invariant names* (think: Java fully qualified class name). They, too, are lists of $\mathbb{N}$, the same type as namespaces. In order to connect this up to the definitions of §8.2, we need a way to map structued invariant names to *InvName*, the type of "plain" invariant names. Any injective mapping $\mathsf{namesp\_inj}$ will do; and such a mapping has to exist because $\mathit{List}(\mathbb{N})$ is countable and *InvName* is infinite. Whenever needed, we (usually implicitly) coerce $\mathcal{N}$ to its encoded suffix-closure, *i.e.,* to the set of encoded structured invariant names contained in the namespace:

$$\mathcal{N}^{\uparrow} \triangleq \{\iota \mid \exists \mathcal{N}'.\, \iota = \mathsf{namesp\_inj}(\mathcal{N}' +\!\!+ \mathcal{N})\}$$

We will overload the notation for invariant assertions for using namespaces instead of names:

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

We can now derive the following rules (this involves unfolding the definition of fancy updates):

INV-PERSIST
$$\boxed{P}^{\mathcal{N}} \vdash \Box \boxed{P}^{\mathcal{N}}$$

INV-ALLOC
$$\triangleright P \vdash \Rrightarrow_{\emptyset} \boxed{P}^{\mathcal{N}}$$

INV-OPEN
$$\frac{\mathcal{N} \subseteq \mathcal{E}}{\boxed{P}^{\mathcal{N}} \;\; \mathcal{E} \Rrightarrow^{\mathcal{E}\setminus\mathcal{N}} \triangleright P * (\triangleright P \;^{\mathcal{E}\setminus\mathcal{N}}\!\!\Rrightarrow\!\!\!\!*^{\mathcal{E}}\; \mathsf{True})}$$

INV-OPEN-TIMELESS
$$\frac{\mathcal{N} \subseteq \mathcal{E} \qquad \mathsf{timeless}(P)}{\boxed{P}^{\mathcal{N}} \;\; \mathcal{E} \Rrightarrow^{\mathcal{E}\setminus\mathcal{N}} P * (P \;^{\mathcal{E}\setminus\mathcal{N}}\!\!\Rrightarrow\!\!\!\!*^{\mathcal{E}}\; \mathsf{True})}$$

8.5 Accessors

The two rules INV-OPEN and INV-OPEN-TIMELESS above may look a little surprising, in the sense that it is not clear on first sight how they would be applied. The rules are the first *accessors* that show up in this document. Accessors are assertions of the form

$$P \ {}^{\mathcal{E}_1}\!\Rrightarrow^{\mathcal{E}_2} \exists x.\, Q * (\forall y.\, Q' \ {}^{\mathcal{E}_2}\!\Rrightarrow\!\!\!\!*^{\mathcal{E}_1} \ R)$$

One way to think about such assertions is as follows: Given some accessor, if during our verification we have the assertion P and the mask $\mathcal{E}_1$ available, we can use the accessor to *access* Q and obtain the witness x. We call this *opening* the accessor, and it changes the mask to $\mathcal{E}_2$. Additionally, opening the accessor provides us with $\forall y.\, Q' \ {}^{\mathcal{E}_2}\!\Rrightarrow\!\!\!\!*^{\mathcal{E}_1} \ R$, a *linear view shift* (*i.e.,* a view shift that can only be used once). This linear view shift tells us that in order to *close* the accessor again and go back to mask $\mathcal{E}_1$, we have to pick some y and establish the corresponding Q'. After closing, we will obtain R.

Using VS-TRANS and HT-ATOMIC (or the corresponding proof rules for fancy updates and weakest preconditions), we can show that it is possible to open an accessor around any view shift and any *atomic* expression:

$$\text{ACC-VS}\ \frac{P \ {}^{\mathcal{E}_1}\!\Rrightarrow^{\mathcal{E}_2} \exists x.\, Q * (\forall y.\, Q' \ {}^{\mathcal{E}_2}\!\Rrightarrow\!\!\!\!*^{\mathcal{E}_1} \ R) \qquad \forall x.\, Q * P_F \Rrightarrow_{\mathcal{E}_2} \exists y.\, Q' * P_F}{P * P_F \Rrightarrow_{\mathcal{E}_1} R * P_F}$$

$$\text{ACC-HT}\ \frac{P \ {}^{\mathcal{E}_1}\!\Rrightarrow^{\mathcal{E}_2} \exists x.\, Q * (\forall y.\, Q' \ {}^{\mathcal{E}_2}\!\Rrightarrow\!\!\!\!*^{\mathcal{E}_1} \ R) \qquad \forall x.\, \{Q * P_F\}\, e\, \{\exists y.\, Q' * P_F\}_{\mathcal{E}_2} \qquad \mathsf{atomic}(e)}{\{P * P_F\}\, e\, \{R * P_F\}_{\mathcal{E}_1}}$$

Furthermore, in the special case that $\mathcal{E}_1 = \mathcal{E}_2$, the accessor can be opened around *any* expression. For this reason, we also call such accessors *non-atomic*.

The reasons accessors are useful is that they let us talk about "opening X" (*e.g.,* "opening invariants") without having to care what X is opened around. Furthermore, as we construct more sophisticated and more interesting things that can be opened (*e.g.,* invariants that can be "cancelled", or STSs), accessors become a useful interface that allows us to mix and match different abstractions in arbitrary ways.

For the special case that $P = R$ and $Q = Q'$, we use the following notation that avoids repetition:

$$\langle P \Longleftrightarrow x.\, Q \rangle^{\mathcal{E}_2}_{\mathcal{E}_1} \triangleq P \ {}^{\mathcal{E}_1}\!\Rrightarrow^{\mathcal{E}_2} \exists x.\, Q * (Q \ {}^{\mathcal{E}_2}\!\Rrightarrow\!\!\!\!*^{\mathcal{E}_1} \ P)$$

This accessor is "idempotent" in the sense that it doesn't actually change the state. After applying it, we get our P back so we end up where we started.

9 Derived constructions

9.1 Non-atomic ("thread-local") invariants

Sometimes it is necessary to maintain invariants that we need to open non-atomically. Clearly, for this mechanism to be sound we need something that prevents us from opening the same invariant twice, something like the masks that avoid reentrancy on the "normal", atomic invariants. The idea is to use tokens[3] that guard access to non-atomic invariants. Having the token $[\text{NaInv} : p.\mathcal{E}]$ indicates that we can open all invariants in $\mathcal{E}$. The p here is the name of the *invariant pool*. This mechanism allows us to have multiple, independent pools of invariants that all have their own namespaces.

One way to think about non-atomic invariants is as "thread-local invariants", where every pool is a thread. Every thread thus has its own, independent set of invariants. Every thread threads through all the tokens for its own pool, so that each invariant can only be opened in the thread it belongs to. As a consequence, they can be kept open around any sequence of expressions (*i.e.,* there is no restriction to atomic expressions) – after all, there cannot be any races with other threads.

Concretely, this is the monoid structure we need:

$$\begin{aligned} \mathit{PId} &\triangleq \mathit{GName} \\ \textsc{NaTok} &\triangleq \wp^{\text{fin}}(\mathit{InvName}) \times \wp(\mathit{InvName}) \end{aligned}$$

For every pool, there is a set of tokens designating which invariants are *enabled* (closed). This corresponds to the mask of "normal" invariants. We re-use the structure given by namespaces for non-atomic invariants. Furthermore, there is a *finite* set of invariants that is *disabled* (open).

Owning tokens is defined as follows:

$$\begin{aligned} [\text{NaInv} : p.\mathcal{E}] &\triangleq \boxed{(\emptyset, \mathcal{E})}^{p} \\ [\text{NaInv} : p] &\triangleq [\text{NaInv} : p.\top] \end{aligned}$$

Next, we define non-atomic invariants. To simplify this construction,we piggy-back into "normal" invariants.

$$\mathsf{NaInv}^{p.\mathcal{N}}(P) \triangleq \exists \iota \in \mathcal{N}. \boxed{P * \boxed{(\{\iota\}, \emptyset)}^{p} \vee [\text{NaInv} : p.\{\iota\}]}^{\mathcal{N}}$$

We easily obtain:

$$\mathsf{True} \Rrightarrow_{\perp} \exists p. [\text{NaInv} : p] \qquad [\text{NaInv} : p.\mathcal{E}_1 \uplus \mathcal{E}_2] \Leftrightarrow [\text{NaInv} : p.\mathcal{E}_1] * [\text{NaInv} : p.\mathcal{E}_2]$$

$$\triangleright P \Rrightarrow_{\mathcal{N}} \Box \mathsf{NaInv}^{p.\mathcal{N}}(P) \qquad \mathsf{NaInv}^{p.\mathcal{N}}(P) \vdash \langle [\text{NaInv} : p.\mathcal{N}] \Longleftrightarrow \triangleright P \rangle_{\mathcal{N}}$$

from which we can derive

$$\frac{\mathcal{N} \subseteq \mathcal{E}}{\mathsf{NaInv}^{p.\mathcal{N}}(P) \vdash \langle [\text{NaInv} : p.\mathcal{E}] \Longleftrightarrow \triangleright P * [\text{NaInv} : p.\mathcal{E} \setminus \mathcal{N}] \rangle_{\mathcal{N}}}$$

[3] Very much like the tokens that are used to encode "normal", atomic invariants

9.2 Boxes

The idea behind the *boxes* is to have an assertion P that is actually split into a number of pieces, each of which can be taken out and back in separately. In some sense, this is a replacement for having an "authoritative PCM of Iris assertions itself". It is similar to the pattern involving saved propositions that was used for the barrier [5], but more complicated because there are some operations that we want to perform without a later.

Roughly, the idea is that a *box* is a container for an assertion P. A box consists of a bunch of *slices* which decompose P into a separating conjunction of the assertions Q_i governed by the individual slices. Each slice is either *full* (it right now contains Q_i), or *empty* (it does not contain anything currently). The assertion governing the box keeps track of the state of all the slices that make up the box. The crux is that opening and closing of a slice can be done even if we only have ownership of the boxes "later" ($\triangleright$).

The interface for boxes is as follows: The two core assertions are: $\mathsf{BoxSlice}(\mathcal{N}, P, i)$, saying that there is a slice in namespace $\mathcal{N}$ with name i and content P; and $\mathsf{Box}(\mathcal{N}, P, f)$, saying that f describes the slices of a box in namespace $\mathcal{N}$, such that all the slices together contain P. Here, f is of type $\mathbb{N} \xrightarrow{\text{fin}} \mathsf{BoxState}$ mapping names to states, where $\mathsf{BoxState} \triangleq \{\mathsf{full}, \mathsf{empty}\}$.

Box-create
$$\mathsf{True} \Rrightarrow_{\mathcal{N}} \mathsf{Box}(\mathcal{N}, \mathsf{True}, \emptyset)$$

Slice-insert-empty
$$\triangleright^b \mathsf{Box}(\mathcal{N}, P, f) \Rrightarrow_{\mathcal{N}} \exists i \notin \mathrm{dom}(f).\ \square\, \mathsf{BoxSlice}(\mathcal{N}, Q, i) * \triangleright^b \mathsf{Box}(\mathcal{N}, P * Q, f\,[i \leftarrow \mathsf{empty}])$$

Slice-delete-empty
$$\frac{f(i) = \mathsf{empty}}{\mathsf{BoxSlice}(\mathcal{N}, Q, i) \vdash \triangleright^b \mathsf{Box}(\mathcal{N}, P, f) \Rrightarrow_{\mathcal{N}} \exists P'.\ \triangleright^b(\triangleright(P = P' * Q) * \mathsf{Box}(\mathcal{N}, P', f\,[i \leftarrow \bot]))}$$

Slice-fill
$$\frac{f(i) = \mathsf{empty}}{\mathsf{BoxSlice}(\mathcal{N}, Q, i) \vdash \triangleright^b Q * \triangleright \mathsf{Box}(\mathcal{N}, P, f) \Rrightarrow_{\mathcal{N}} \triangleright^b \mathsf{Box}(\mathcal{N}, P, f\,[i \leftarrow \mathsf{full}])}$$

Slice-empty
$$\frac{f(i) = \mathsf{full}}{\mathsf{BoxSlice}(\mathcal{N}, Q, i) \vdash \triangleright^b \mathsf{Box}(\mathcal{N}, P, f) \Rrightarrow_{\mathcal{N}} \triangleright Q * \triangleright^b \mathsf{Box}(\mathcal{N}, P, f\,[i \leftarrow \mathsf{empty}])}$$

Box-fill
$$\frac{\forall i \in \mathrm{dom}(f).\ f(i) = \mathsf{empty}}{\triangleright P * \mathsf{Box}(\mathcal{N}, P, f) \Rrightarrow_{\mathcal{N}} \mathsf{Box}(\mathcal{N}, P, f\,[i \leftarrow \mathsf{full} \mid i \in \mathrm{dom}(f)])}$$

Box-empty
$$\frac{\forall i \in \mathrm{dom}(f).\ f(i) = \mathsf{full}}{\mathsf{Box}(\mathcal{N}, P, f) \Rrightarrow_{\mathcal{N}} \triangleright P * \mathsf{Box}(\mathcal{N}, P, f\,[i \leftarrow \mathsf{empty} \mid i \in \mathrm{dom}(f)])}$$

Above, $\triangleright^b P$ is syntactic sugar for $\triangleright P$ (if b is 1) or P (if b is 0). This is essentially an *optional later*, indicating that the lemmas can be applied with Box being owned now or later, and that ownership is returned the same way.

Model. The above rules are validated by the following model. We need a CMRA as follows:

$$\mathsf{BoxState} \triangleq \mathsf{full} + \mathsf{empty}$$
$$\textsc{Box} \triangleq \textsc{Auth}(\textsc{Ex}(\mathsf{BoxState})^?) \times \textsc{Ag}(\blacktriangleright \mathit{iProp})^?$$

Now we can define the assertions:

$$\mathsf{SliceInv}(i, P) \triangleq \exists b. \boxed{(\bullet b, \varepsilon)}^i * ((b = \mathsf{full}) \Rightarrow P)$$
$$\mathsf{BoxSlice}(\mathcal{N}, P, i) \triangleq \boxed{(\varepsilon, P)}^i * \boxed{\mathsf{SliceInv}(i, P)}^{\mathcal{N}}$$
$$\mathsf{Box}(\mathcal{N}, P, f) \triangleq \exists Q : \mathbb{N} \to \mathsf{iProp}.\ \triangleright \left(P = \ast_{i \in \mathrm{dom}(f)} Q(i)\right) * \ast_{i \in \mathrm{dom}(f)} \boxed{(\circ f(i), Q(i))}^i * \boxed{\mathsf{SliceInv}(i, Q(i))}^{\mathcal{N}}$$

Derived rules. Here are some derived rules:

Slice-insert-full

$$\triangleright Q * \triangleright^b \mathsf{Box}(\mathcal{N}, P, f) \Rrightarrow_{\mathcal{N}} \exists i \notin \mathrm{dom}(f).\ \square\, \mathsf{BoxSlice}(\mathcal{N}, Q, i) * \triangleright^b \mathsf{Box}(\mathcal{N}, P * Q, f\,[i \leftarrow \mathsf{full}])$$

Slice-delete-full

$$\frac{f(i) = \mathsf{full}}{\mathsf{BoxSlice}(\mathcal{N}, Q, i) \vdash \triangleright^b \mathsf{Box}(\mathcal{N}, P, f) \Rrightarrow_{\mathcal{N}} \triangleright Q * \exists P'.\ \triangleright^b(\triangleright(P = P' * Q) * \mathsf{Box}(\mathcal{N}, P', f\,[i \leftarrow \bot]))}$$

Slice-split

$$\frac{f(i) = s}{\begin{array}{l}\mathsf{BoxSlice}(\mathcal{N}, Q_1 * Q_2, i) \vdash \triangleright^b \mathsf{Box}(\mathcal{N}, P, f) \Rrightarrow_{\mathcal{N}} \exists i_1 \notin \mathrm{dom}(f), i_2 \notin \mathrm{dom}(f).\ i_1 \neq i_2 \wedge \\ \quad \square\, \mathsf{BoxSlice}(\mathcal{N}, Q_1, i_1) * \square\, \mathsf{BoxSlice}(\mathcal{N}, Q_2, i_2) * \triangleright^b \mathsf{Box}(\mathcal{N}, P, f\,[i \leftarrow \bot]\,[i_1 \leftarrow s]\,[i_2 \leftarrow s])\end{array}}$$

Slice-merge

$$\frac{i_1 \neq i_2 \qquad f(i_1) = f(i_2) = s}{\begin{array}{l}\mathsf{BoxSlice}(\mathcal{N}, Q_1, i_1), \mathsf{BoxSlice}(\mathcal{N}, Q_2, i_2) \vdash \triangleright^b \mathsf{Box}(\mathcal{N}, P, f) \Rrightarrow_{\mathcal{N}} \exists i \notin \mathrm{dom}(f) \setminus \{i_1, i_2\}.\ \\ \quad \square\, \mathsf{BoxSlice}(\mathcal{N}, Q_1 * Q_2, i) * \triangleright^b \mathsf{Box}(\mathcal{N}, P, f\,[i_1 \leftarrow \bot]\,[i_2 \leftarrow \bot]\,[i \leftarrow s])\end{array}}$$

10 Logical paradoxes

In this section we provide proofs of some logical inconsistencies that arise when slight changes are made to the Iris logic.

10.1 Saved propositions without a later

As a preparation for the proof about invariants in §10.2, we show that omitting the later modality from a variant of *saved propositions* leads to a contradiction. Saved propositions have been introduced in prior work [4, 5] to prove correctness of synchronization primitives; we will explain all that is necessary here. The counterexample assumes a higher-order logic with separating conjunction, magic wand and the modalities $\Box$ and $\Rrightarrow$ satisfying the rules in §4.

Theorem 1. *If there exists a type GName and an assertion* $\_ \Mapsto \_ : \mathit{GName} \to \mathit{iProp} \to \mathit{iProp}$ *associating names* $\gamma : \mathit{GName}$ *to propositions and satisfying:*

$$\vdash \Rrightarrow \exists \gamma : \mathit{GName}.\, \gamma \Mapsto P(\gamma) \qquad \text{(SPROP-ALLOC)}$$

$$\gamma \Mapsto P \vdash \Box(\gamma \Mapsto P) \qquad \text{(SPROP-PERSIST)}$$

$$\gamma \Mapsto P * \gamma \Mapsto Q \vdash P \Leftrightarrow Q \qquad \text{(SPROP-AGREE)}$$

then $\vdash \Rrightarrow \mathit{False}$.

The type *GName* should be thought of as the type of "locations" and $\gamma \Mapsto P$ should be read as stating that location γ "stores" proposition P. Notice that these are immutable locations, so the maps-to assertion is persistent. The rule SPROP-ALLOC is then thought of as allocation, and the rule SPROP-AGREE states that a given location γ can only store *one* proposition, so multiple witnesses covering the same location must agree.

The conclusion of SPROP-AGREE usually is guarded by a $\triangleright$. The point of this theorem is to show that said later is *essential*, as removing it introduces inconsistency. The key to proving Theorem 1 is the following assertion:

Definition 23. $A(\gamma) \triangleq \exists P : \mathit{iProp}.\, \Box \neg P \land \gamma \Mapsto P$.

Intuitively, $A(\gamma)$ says that the saved proposition named γ does *not* hold, *i.e.,* we can disprove it. Using SPROP-PERSIST, it is immediate that $A(\gamma)$ is persistent.

Now, by applying SPROP-ALLOC with A, we obtain a proof of $P \triangleq \gamma \Mapsto A(\gamma)$: this says that the proposition named γ is the assertion saying that it, itself, doesn't hold. In other words, P says that the assertion named γ expresses its own negation. Unsurprisingly, that leads to a contradiction, as is shown in the following lemma:

Lemma 2. *We have* $\gamma \Mapsto A(\gamma) \vdash \Box \neg A(\gamma)$ *and* $\gamma \Mapsto A(\gamma) \vdash A(\gamma)$.

Proof.

- First we show $\gamma \Mapsto A(\gamma) \vdash \Box \neg A(\gamma)$. Since $\gamma \Mapsto A(\gamma)$ is persistent it suffices to show $\gamma \Mapsto A(\gamma) \vdash \neg A(\gamma)$. Suppose $\gamma \Mapsto A(\gamma)$ and $A(\gamma)$. Then by definition of A there is a P such that $\Box \neg P$ and $\gamma \Mapsto P$. By SPROP-AGREE we have $P \Leftrightarrow A(\gamma)$ and so from $\neg P$ we get $\neg A(\gamma)$, which leads to a contradiction with $A(\gamma)$.

- Using the first item we can now prove $\gamma \Rrightarrow A(\gamma) \vdash A(\gamma)$. We need to prove

$$\exists P : \mathsf{iProp}.\ \Box \neg P \land \gamma \Rrightarrow P.$$

  We do so by picking P to be $A(\gamma)$, which leaves us to prove $\Box \neg A(\gamma) \land \gamma \Rrightarrow A(\gamma)$. The last conjunct holds by assumption, and the first conjunct follows from the previous item of this lemma.

□

With this lemma in hand, the proof of Theorem 1 is simple.

Theorem 1. Using the previous lemmas we have

$$\vdash \forall \gamma. \neg(\gamma \Rrightarrow A(\gamma)).$$

Together with the rule SPROP-ALLOC we thus derive $\Rrightarrow \mathsf{False}$. □

10.2 Invariants without a later

Now we come to the main paradox: if we remove the $\triangleright$ from INV-OPEN, the logic becomes inconsistent. The theorem is stated as general as possible so taht it also applies to previous, less powerful versions of Iris.

Theorem 2. *Assume a higher-order separation logic with $\Box$ and an update modality with a binary mask $\Rrightarrow_{\{0,1\}}$ (think: empty mask and full mask) satisfying strong monad rules with respect to separating conjunction and such that:*

$$\text{WEAKEN-MASK}\quad \Rrightarrow_0 P \vdash \Rrightarrow_1 P$$

Assume a type InvName and an assertion $\boxed{\cdot}^{\cdot} : \mathit{InvName} \to \mathit{iProp} \to \mathit{iProp}$ satisfying:

$$\text{INV-ALLOC}\quad P \vdash \Rrightarrow_1 \exists \iota. \boxed{P}^{\iota} \qquad \text{INV-PERSIST}\quad \boxed{P}^{\iota} \vdash \Box \boxed{P}^{\iota} \qquad \text{INV-OPEN-NOLATER}\quad \frac{P * Q \vdash \Rrightarrow_0 (P * R)}{\boxed{P}^{\iota} * Q \vdash \Rrightarrow_1 R}$$

Finally, assume the existence of a type GName and two tokens $\boxed{\mathrm{S}}^{\cdot} : \mathit{GName} \to \mathit{iProp}$ and $\boxed{\mathrm{F}}^{\cdot} : \mathit{GName} \to \mathit{iProp}$ parameterized by GName and satisfying the following properties:

$$\text{START-ALLOC}\quad \vdash \Rrightarrow_0 \exists \gamma. \boxed{\mathrm{S}}^{\gamma} \qquad \text{START-FINISH}\quad \boxed{\mathrm{S}}^{\gamma} \vdash \Rrightarrow_0 \boxed{\mathrm{F}}^{\gamma} \qquad \text{START-NOT-FINISHED}\quad \boxed{\mathrm{S}}^{\gamma} * \boxed{\mathrm{F}}^{\gamma} \vdash \mathit{False} \qquad \text{FINISHED-DUP}\quad \boxed{\mathrm{F}}^{\gamma} \vdash \boxed{\mathrm{F}}^{\gamma} * \boxed{\mathrm{F}}^{\gamma}$$

Then $\mathit{True} \vdash \Rrightarrow_1 \mathit{False}$.

The core of the proof is defining the $\Rrightarrow$ from the previous counterexample using invariants. Then, using the standard proof rules for invariants, we show that it satisfies SPROP-ALLOC and SPROP-PERSIST. Furthermore, assuming the rule for opening invariants without a $\triangleright$, we can prove a slightly weaker version of SPROP-AGREE, which is sufficient for deriving a contradiction.

We start by defining $\Rrightarrow$ satisfying (almost) the assumptions of Lemma 4.

Definition 24. *We define* $\_ \Rrightarrow \_ : \mathit{GName} \to \mathit{iProp} \to \mathit{iProp}$ *as:*

$$\gamma \Rrightarrow P \triangleq \exists \iota. \boxed{\boxed{\mathrm{S}}^{\gamma} \vee \boxed{\mathrm{F}}^{\gamma} * \Box P}^{\iota}.$$

Note that using INV-PERSIST, it is immediate that $\gamma \Rrightarrow P$ is persistent.

We use the tokens $\boxed{\mathrm{S}}^{\gamma}$ and $\boxed{\mathrm{F}}^{\gamma}$ to model invariants that can be initialized "lazily": $\boxed{\mathrm{S}}^{\gamma}$ indicates that the invariant is still not initialized, whereas the duplicable $\boxed{\mathrm{F}}^{\gamma}$ indicates it has been initialized with a resource satisfying P.

We can show variants of SPROP-AGREE and SPROP-ALLOC for the defined $\Rrightarrow$.

Lemma 3. *We have* $\vdash \Rrightarrow_{\top} \exists \gamma. \gamma \Rrightarrow P(\gamma)$.

Proof. We have to show the allocation rule

$$\vdash \Rrightarrow_{\top} \exists \gamma. \gamma \Rrightarrow P.$$

From START-ALLOC we have a γ such that $\Rrightarrow_{\bot} \boxed{\mathrm{S}}^{\gamma}$ holds and hence from WEAKEN-MASK we have $\Rrightarrow_{\top} \boxed{\mathrm{S}}^{\gamma}$. Since we are proving a goal of the form $\Rrightarrow_{\top} R$ we may assume $\boxed{\mathrm{S}}^{\gamma}$. Thus for any P we have $\Rrightarrow_{\top} \left(\boxed{\mathrm{S}}^{\gamma} \vee \boxed{\mathrm{F}}^{\gamma} * P\right)$. Again since our goal is still of the form $\Rrightarrow_{\top}$ we may assume $\boxed{\mathrm{S}}^{\gamma} \vee \boxed{\mathrm{F}}^{\gamma} * \Box P$. The rule INV-ALLOC then gives us precisely what we need. □ □

Lemma 4. *We have* $\gamma \Rrightarrow P * \gamma \Rrightarrow Q * \Box P \vdash \Rrightarrow_{\top} \Box Q$ *and thus* $\gamma \Rrightarrow P * \gamma \Rrightarrow Q \vdash (\Rrightarrow_{\top} \Box P) \Leftrightarrow (\Rrightarrow_{\top} \Box Q)$.

Lemma 4.

- We first show

$$\gamma \Rrightarrow P * \gamma \Rrightarrow Q * \Box P \vdash \Rrightarrow_{\top} \Box Q.$$

We use INV-OPEN-NOLATER to open the invariant in $\gamma \Rrightarrow P$ and consider two cases:

1. $\boxed{\mathrm{S}}^{\gamma}$(the invariant is "uninitialized") : In this case, we use START-FINISH to "initialize" the invariant and obtain $\boxed{\mathrm{F}}^{\gamma}$. Then we duplicate $\boxed{\mathrm{F}}^{\gamma}$, and use it together with $\Box P$ to close the invariant.
2. $\boxed{\mathrm{F}}^{\gamma} * \Box P$ (the invariant is "initialized"): In this case we duplicate $\boxed{\mathrm{F}}^{\gamma}$, and use a copy to close the invariant.

After closing the invariant, we have obtained $\boxed{\mathrm{F}}^{\gamma}$. Hence, it is sufficient to prove

$$\boxed{\mathrm{F}}^{\gamma} * \gamma \Rrightarrow P * \gamma \Rrightarrow Q * \Box P \vdash \Rrightarrow_{\top} \Box Q.$$

We proceed by using INV-OPEN-NOLATER to open the other invariant in $\gamma \Rrightarrow Q$, and we again consider two cases:

1. $\boxed{\mathrm{S}}^{\gamma}$ (the invariant is "uninitialized"): As witnessed by START-NOT-FINISHED, this cannot happen, so we derive a contradiction. Notice that this is a key point of the proof: because the two invariants ($\gamma \Rrightarrow P$ and $\gamma \Rrightarrow Q$) *share* the ghost name γ, initializing one of them is enough to show that the other one has been initialized. Essentially, this is an indirect way of saying that really, we have been opening the same invariant two times.
2. $\boxed{\mathrm{F}}^{\gamma} * \Box Q$ (the invariant is "initialized"): Since $\Box Q$ is duplicable we use one copy to close the invariant, and retain another to prove $\Rrightarrow_{\top} \Box Q$.

- By applying the above twice, we easily obtain

$$\gamma \Rightarrow P * \gamma \Rightarrow Q \vdash (\Rrightarrow_\top \Box P) \Leftrightarrow (\Rrightarrow_\top \Box Q)$$

□ □

Intuitively, Lemma 4 shows that we can "convert" a proof from P to Q.

We are now in a position to replay the counterexample from §10.1. The only difference is that because Lemma 4 is slightly weaker than the rule SPROP-AGREE of Theorem 1, we need to use $\Rrightarrow_\top \mathsf{False}$ in place of False in the definition of the predicate A: we let $A(\gamma) \triangleq \exists P : \mathsf{iProp}.\ \Box(P \Rightarrow \Rrightarrow_\top \mathsf{False}) \wedge \gamma \Rightarrow P$ and replay the proof that we have presented above.

References

[1] Pierre America and Jan Rutten. "Solving Reflexive Domain Equations in a Category of Complete Metric Spaces". In: *JCSS* 39.3 (1989), pp. 343–375.

[2] Lars Birkedal and Aleš Bizjak. *A Taste of Categorical Logic — Tutorial Notes*. Available at http://users-cs.au.dk/birke/modures/tutorial/categorical-logic-tutorial-notes.pdf. Oct. 2014.

[3] Lars Birkedal, Kristian Støvring, and Jacob Thamsborg. "The category-theoretic solution of recursive metric-space equations". In: *TCS* 411.47 (2010), pp. 4102–4122. DOI: 10.1016/j.tcs.2010.07.010. URL: http://dx.doi.org/10.1016/j.tcs.2010.07.010.

[4] Mike Dodds et al. "Verifying Custom Synchronization Constructs Using Higher-Order Separation Logic". In: *TOPLAS* 38.2 (2016), p. 4. DOI: 10.1145/2818638. URL: http://doi.acm.org/10.1145/2818638.

[5] Ralf Jung et al. "Higher-order ghost state". In: *ICFP*. 2016, pp. 256–269.

[6] Aaron Turon, Derek Dreyer, and Lars Birkedal. "Unifying refinement and Hoare-style reasoning in a logic for higher-order concurrency". In: *ICFP*. 2013, pp. 377–390.