# Impredicative Concurrent Abstract Predicates
# (Technical Appendix)

Kasper Svendsen
Aarhus University
ksvendsen@cs.au.dk

Lars Birkedal
Aarhus University
birkedal@cs.au.dk

November 1, 2013

## Contents

## Introduction

In this techincal appendix we present the full model of iCAP in the topos of trees, along with a number of meta-theoretic results about the model. We also present a formal proof system for iCAP, along with its interpretation. Using this formal proof system we verify a fine-grained implementation of a concurrent bag, implemented using helping, against a refineable specification. Lastly, we develop a number of technical results about the topos of trees, that we use to reason about the iCAP model.

In Section 1, the ambient meta-theory is the topos of trees and we reason using the internal logic of the topos of trees. Sections 2 and 3 we define the logic and its interpretation in the topos of trees using classical mathematics as our ambient meta-theory. In Section 4, the ambient meta-theory is iCAP. In Section 5, we develop some of the meta-theoretic results internally in the topos of trees, and others we prove externally in classical mathematics, using the Kripke-Joyal forcing semantics. Lastly, in Section 6 we define the formal semantics of a subset of $\text{C}^\sharp$ in classical mathematics. Since the operational semantics is simply an inductive definition (expressible as a W-type) and the constant sets functor preserves W-types, we can freely used this operational semantics *inside* the topos of trees in Section 1.

## 1 Model

In this section we present the model of iCAP. The non-recursive parts of the model are defined in the category of Sets, Sets, and the recursive parts in the topos of trees, $\mathcal{S}$.

*Local and shared states* $\boxed{\mathit{LState}, \mathit{LTS}, \mathit{SState} \in \text{Sets}}$

Local states consists of a plain heap and a state transition capability map. Shared states consist of an abstract state and a labelled transition system for each shared region.

$$
\begin{aligned}
\mathit{Perm} &\stackrel{\text{def}}{=} \{q \in \mathbb{Q} \mid 0 \leq q \leq 1\} \\
\mathit{Perm}_n &\stackrel{\text{def}}{=} \{q \in \mathbb{Q} \mid 0 < q \leq 1\} \\
\mathit{Cap} &\stackrel{\text{def}}{=} \{f : (\mathit{RId} \times \mathit{AId}) \to \mathit{Perm} \mid \exists R \subseteq_{fin} \mathit{RId}. \\
&\qquad\qquad \forall r \in \mathit{RId} \setminus R.\ \forall \alpha \in \mathit{AId}.\ f(r, \alpha) = 0\} \\
\mathit{Heap} &\stackrel{\text{def}}{=} (\mathit{OId} \times \mathit{FName} \rightharpoonup_{fin} \mathit{CVal}) \times (\mathit{OId} \rightharpoonup_{fin} \mathit{CName}) \\
&\qquad \times (\mathit{CId} \rightharpoonup_{fin} \mathit{OId} \times \mathit{MName}) \\
\mathit{PHeap} &\stackrel{\text{def}}{=} \{(pc, ph) \in (\mathit{OId} \times \mathit{FName} \to \mathit{Perm}) \times (\mathit{OId} \times \mathit{FName} \rightharpoonup_{fin} \mathit{Val}) \mid \\
&\qquad\qquad \forall o \in \mathit{OId}.\ \forall f \in \mathit{FName}.\ pc(o, f) = 0 \Rightarrow (o, f) \notin \mathit{dom}(ph)\} \\
\mathit{LState} &\stackrel{\text{def}}{=} \mathit{PHeap} \times \mathit{Heap} \times \mathit{Cap} \\
\mathit{LTS} &\stackrel{\text{def}}{=} \mathit{AId} \to \mathcal{P}(\mathit{SId} \times \mathit{SId}) \\
\mathit{SState} &\stackrel{\text{def}}{=} \mathit{RId} \rightharpoonup_{fin} (\mathit{SId} \times \mathit{LTS})
\end{aligned}
$$

where $Val$ denotes the least set such that

$$Val \cong CVal \uplus Strings \uplus (Val \times Val)$$

We use $l.h$ and $l.c$ to refer to the heap and capability component of a local state $l \in LState$. We use $s(r).s$ and $s(r).p$ to refer to the state identifier and labelled transition system component of region $r$ of a shared state $s \in SState$.

*Local state composition* $\boxed{\bullet_{LState} : \mathcal{P}(\Delta(LState) \times \Delta(LState) \times \Delta(LState)) \in \mathcal{S}}$

$$\begin{aligned}\bullet_{LState} = \{(l_1, l_2, l_r) \mid\ & dom(l_1.h) \cap dom(l_2.h) = \emptyset \wedge dom(l_r.h) = dom(l_1.h) \cup dom(l_2.h) \\ & \wedge\ (\forall x \in dom(l_1.h).\ l_1.h(x) = l_r.h(x)) \\ & \wedge\ (\forall x \in dom(l_2.h).\ l_2.h(x) = l_r.h(x)) \\ & \wedge\ (\forall t : \Delta(RId \times AId).\ l_1.c(t) + l_2.c(t) \leq 1) \\ & \wedge\ (\forall t : \Delta(RId \times AId).\ l_r.c(t) = l_1.c(t) + l_2.c(t))\}\end{aligned}$$

This relation is functional and thus induces a partial function

$$\bullet_{LState} : \Delta(LState) \times \Delta(LState) \rightharpoonup \Delta(LState) \in \mathcal{S}$$

**Lemma 1.**

$$\forall l_1, l_2, l_3 \in \Delta(LState).\ up(l_1 = l_2 \bullet_{LState} l_3)$$

*Proof.* Using the theory of upwards-closure from Section 5.3. □

*Action allowed* $\boxed{act_{allowed} : LState \times RId \to \mathcal{P}(AId) \in \text{Sets}}$

$$act_{allowed}(l, r) \stackrel{\text{def}}{=} \{\alpha \in AId \mid l.c(r, \alpha) < 1\}$$

*Update allowed* $\boxed{upd_{allowed} : LState \times RId \times LTS \to \mathcal{P}(SId \times SId) \in \text{Sets}}$

$$upd_{allowed}(l, r, p) \stackrel{\text{def}}{=} \{(s_1, s_2) \in SId \times SId \mid \exists \alpha \in act_{allowed}(l, r).\ (s_1, s_2) \in p(\alpha)\}$$

*Abstract state* $\boxed{AState \in \text{Sets}}$

$$AState \stackrel{\text{def}}{=} LState \times SState$$

*Interference relation* $\boxed{R_{(-)} : \mathcal{P}(RId) \to \mathcal{P}(AState \times AState) \in \text{Sets}}$

$$
\begin{array}{l}
(l_1, s_1)\ R_A\ (l_2, s_2) \quad \text{iff} \\
\quad l_1 \leq l_2\ \wedge \\
\quad\quad \forall r \in dom(s_1).\ (\,(r \in A \wedge (s_1(r).s, s_2(r).s) \in \overline{upd_{allowed}(l_1, r, s_1(r).p)}) \vee \\
\quad\quad\quad\quad s_1(r).s = s_2(r).s) \wedge s_1(r).p = s_2(r).p
\end{array}
$$

where $\overline{R}$ denotes the reflexive, transitive closure of $R$ and the ordering on *SState* is given by,

$$s_1 \leq s_2 \stackrel{\text{def}}{=} dom(s_1) \subseteq dom(s_2) \wedge \forall r \in dom(s_1).\ s_1(r) = s_2(r)$$

We use $R$ as shorthand for $R_{RId}$.

*Region interpretations* $\boxed{RIntr \in \mathcal{S}}$

A region interpretation gives the concrete interpretation of each of the abstract shared states for each shared region. Since the state interpretation might itself refer to the state interpretations of regions, the type of state interpretations is recursively defined as a guarded recursive type in the topos of trees, $\mathcal{S}$. Let *RIntr* denote an object in $\mathcal{S}$ satisfying the following isomorphism,

$$i : RIntr \cong \blacktriangleright((\Delta(SId) \times (\Delta(RId) \rightharpoonup_{fin} RIntr)) \to_{mon} \mathcal{P}^{\uparrow}(\Delta(AState)))$$

where $\Delta(AState)$ is upwards-closed with respect to the interference relation $R$, and the ordering on $\Delta(RId) \rightharpoonup_{fin} RIntr$ is given by,

$$\varsigma_1 \leq \varsigma_2 \stackrel{\text{def}}{=} dom(\varsigma_1) \subseteq dom(\varsigma_2) \wedge \forall r \in dom(\varsigma_1).\ \varsigma_1(r) = \varsigma_2(r)$$

and the ordering on $\Delta(SId)$ is the identity.

Define

$$
\begin{array}{l}
lam : (\Delta(SId) \times (\Delta(RId) \rightharpoonup_{fin} RIntr) \to_{mon} \mathcal{P}^{\uparrow}(\Delta(AState))) \to RIntr \\
app : RIntr \to (\Delta(SId) \times (\Delta(RId) \rightharpoonup_{fin} RIntr) \to_{mon} \mathcal{P}^{\uparrow}(\Delta(AState)))
\end{array}
$$

as follows

$$
\begin{array}{l}
lam \stackrel{\text{def}}{=} i^{-1} \circ next \\
app \stackrel{\text{def}}{=}\ \lambda x : RIntr.\ \lambda(y, z) \in \Delta(SId)) \times (\Delta(RId) \rightharpoonup_{fin} RIntr).\ \lambda a : \Delta(AState). \\
\quad\quad succ((J(i(x))(next(y), next(z)))(next(a)))
\end{array}
$$

where $J_{X,Y}$ refers to the morphism $\blacktriangleright(X \to Y) \to \blacktriangleright X \to \blacktriangleright Y$ from (2.1) in [1].

**Lemma 2.**

$$app \circ lam = \triangleright$$

*Instrumented states* $\boxed{\mathcal{M} \in \mathcal{S}}$

Instrumented states thus consist of a concrete local state, an abstract state transition systems for each shared region, and concrete interpretations for each abstract shared state.

$$\mathcal{M} \stackrel{\text{def}}{=} \Delta(\mathit{LState}) \times \Delta(\mathit{SState}) \times \mathit{AMod}$$

where $\mathit{AMod} = \Delta(\mathit{RId}) \rightharpoonup_{\mathit{fin}} \mathit{RIntr}$. We use $m.l$, $m.s$, and $m.a$ to refer to the local state, the shared state and the action model component of an instrumented state $m \in \mathcal{M}$.

*Prop* $\boxed{\mathit{Prop} \in \mathcal{S}}$

Assertions are upwards-closed subsets of instrumented states.

$$\mathit{Prop} \stackrel{\text{def}}{=} \{p \in \mathcal{P}(\mathcal{M}) \mid \forall m_1 \in p.\ \forall m_2 \in \mathcal{M}.\ m_1 \leq m_2 \Rightarrow m_2 \in p\}$$

where the ordering on $\mathcal{M}$ is given by:

$$m_1 \leq m_2 \stackrel{\text{def}}{=} m_1.l \leq m_2.l \wedge m_1.s \leq m_2.s \wedge m_1.a \leq m_2.a$$

*Region erasure* $\boxed{\lfloor - \rfloor_{(=)} : (\Delta(\mathit{SState}) \times \mathit{AMod}) \times \Delta(\mathit{RId}) \to \mathcal{P}(\Delta(\mathit{LState})) : \mathcal{S}}$

$$\lfloor (s, \varsigma) \rfloor_r \stackrel{\text{def}}{=} \{l \in \mathit{LState} \mid (l, s) \in \mathit{app}(\varsigma(r))(s(r).s, \varsigma)\}$$

*Erasure* $\boxed{\lfloor - \rfloor_{(=)} : \mathcal{M} \times \mathcal{P}(\Delta(\mathit{RId})) \to \mathcal{P}(\Delta(\mathit{Heap})) : \mathcal{S}}$

$$\begin{aligned}\lfloor (l, s, \varsigma) \rfloor_A \stackrel{\text{def}}{=} \{h \in \mathit{Heap} \mid\ & \exists l', sr : \mathit{dom}(s) \cap A \to \mathit{LState}. \\ & h = l'.h \wedge l' = l \bullet \Pi_{r \in \mathit{dom}(s) \cap A} sr(r) \ \wedge \\ & \forall r \in \mathit{dom}(s) \cap A.\ sr(r) \in \lfloor (s, \varsigma) \rfloor_r\}\end{aligned}$$

We use $\lfloor m \rfloor$ as shorthand for $\lfloor m \rfloor_{\mathit{RId}}$.

*Composition* $\boxed{\bullet_{\mathcal{M}} : \mathcal{M} \times \mathcal{M} \rightharpoonup \mathcal{M}}$

$$\bullet_{\mathcal{M}} = \bullet_{\mathit{LState}} \times \bullet_{=} \times \bullet_{=}$$

*Empty resource* $\boxed{\mathit{emp} : \mathit{Prop}}$

$$\mathit{emp} \stackrel{\text{def}}{=} \mathcal{M}$$

*Separating Conjunction* $\boxed{* : \mathit{Prop} \times \mathit{Prop} \to \mathit{Prop} \in \mathcal{S}}$

$$p * q \stackrel{\text{def}}{=} \{m \in \mathcal{M} \mid \exists m_1, m_2 \in \mathcal{M}.\ m = m_1 \bullet m_2 \wedge m_1 \in p \wedge m_2 \in q\}$$

*Implication* $\boxed{\Rightarrow : \mathit{Prop} \times \mathit{Prop} \to \mathit{Prop} \in \mathcal{S}}$

$$p \Rightarrow q \stackrel{\text{def}}{=} \{m \in \mathcal{M} \mid \forall m' \geq m.\ m' \in p \Rightarrow m' \in q\}$$

*Quantifiers* $\boxed{\exists_\tau, \forall_\tau : (\tau \to \mathit{Prop}) \to \mathit{Prop} \in \mathcal{S}}$

$$\exists_\tau(p) \stackrel{\text{def}}{=} \{m \in \mathcal{M} \mid \exists x : \tau.\ m \in p(x)\}$$
$$\forall_\tau(p) \stackrel{\text{def}}{=} \{m \in \mathcal{M} \mid \forall x : \tau.\ m \in p(x)\}$$

**Lemma 3.** *Let $X$ be a total and inhabited object in $\mathcal{S}$. Then*

$$\forall p \in X \to \mathit{Prop}.\ \triangleright \exists_X(p) \Leftrightarrow \exists_X(\lambda x.\, \triangleright p(x))$$

*Proof.* Let $m \in \mathcal{M}$ such that $m \in \triangleright \exists_X(p)$. Then $\triangleright(m \in \exists_X(p)$ and thus

$$\triangleright(\exists x : X.\ m \in p(x))$$

By totality and inhabitance, $\exists x : X.\ \triangleright\, (m \in p(x))$ and thus $m \in \exists_X(\lambda x.\, \triangleright\, p(x))$.

Likewise, let $m \in \mathcal{M}$ such that $m \in \exists_X(\lambda x.\, \triangleright\, p(x))$ then

$$\exists x : X.\ \triangleright\, (m \in p(x))$$

and thus $\triangleright(\exists x : X.\ m \in p(x))$. □

*Validity* $\boxed{\mathit{valid} : \mathcal{P}(\mathit{Prop}) \in \mathcal{S}}$

$$\mathit{valid}(p) \stackrel{\text{def}}{=} \forall m \in \mathcal{M}.\ m \in p$$

**Lemma 4.**

$$R(\mathit{emp}) \subseteq \mathit{emp} \wedge \forall p, q \in \mathit{Prop}.\ R(p * q) \subseteq R(p) * R(q)$$

*Region assertion* $\boxed{\mathit{region} : \mathcal{P}(\Delta(\mathit{SId})) \times \Delta(\mathit{LTS}) \times \Delta(\mathit{RId}) \to \mathit{Prop} \in \mathcal{S}}$

$$\mathit{region}(X, p, r) \stackrel{\text{def}}{=} \{(l, s, \varsigma) \in \mathcal{M} \mid s(r).s \in X \wedge s(r).p = p\}$$

*Region interpretation* $\boxed{rintr : (\Delta(SId) \to Prop) \times \Delta(RId) \to Prop \in \mathcal{S}}$

$$rintr(I, r) \stackrel{\text{def}}{=} \{(l, s, \varsigma) \in \mathcal{M} \mid r \in dom(\varsigma) \wedge \forall x \in \Delta(SId).\ \forall \varsigma' \geq \varsigma.\ app(\varsigma(r))(x, \varsigma') = \triangleright(\lambda(l, s).\ I(x)(l, s, \varsigma'))\}$$

*Action permission* $\boxed{[-]^{(=)}_{(\equiv)} : \Delta(AId) \times \Delta(RId) \times \Delta(Perm_n) \to Prop \in \mathcal{S}}$

$$[\alpha]^r_\pi \stackrel{\text{def}}{=} \{(l, s, \varsigma) \in \mathcal{M} \mid \pi \leq l.c(r, \alpha)\}$$

*Stability* $\boxed{stable_{(-)} : \mathcal{P}(\Delta(RId)) \to \mathcal{P}(Prop) \in \mathcal{S}}$

$$stable_A(p) \stackrel{\text{def}}{=} R_A(p) \subseteq p$$

We use *stable* as shorthand for $stable_{RId}$.

**Lemma 5.**

$$\forall A \in \mathcal{P}(\Delta(RId)).\ \forall p \in Prop.\ stable_A(p) \Rightarrow stable_A(\triangleright p)$$

*Proof.* Assume $m_1 \in \triangleright p$ and $m_1\ R_A\ m_2$. By assumption, $R_A(p) \subseteq p$, and thus, by monotonicity of $\triangleright$,

$$\triangleright(\forall m_1, m_2 \in \mathcal{M}.\ m_1 \in p \wedge m_1\ R_A\ m_2 \Rightarrow m_2 \in p)$$

Thus, since $\mathcal{M}$ is total and inhabited,

$$\forall m_1, m_2 \in \mathcal{M}.\ m_1 \in \triangleright p \wedge \triangleright(m_1\ R_A\ m_2) \Rightarrow m_2 \in \triangleright p$$

and thus, $m_2 \in \triangleright p$. □

*View shift* $\boxed{\sqsubseteq_{(-)} : \mathcal{P}(\Delta(RId)) \to \mathcal{P}(Prop \times Prop) \in \mathcal{S}}$

$$p \sqsubseteq_A q \stackrel{\text{def}}{=} \forall r \in Prop.\ stable_A(r) \Rightarrow \lfloor p * r \rfloor_A \subseteq \lfloor q * r \rfloor_A$$

We use $\sqsubseteq$ as shorthand for $\sqsubseteq_{RId}$.

**Lemma 6.**

$$\forall p, q, r \in Prop.\ stable(r) \wedge p \sqsubseteq_A q \Rightarrow p * r \sqsubseteq_A q * r$$

**Lemma 7.**

$$\forall p_1, p_2, q_1, q_2 \in Prop.\ p_1 \sqsubseteq q_1 \wedge p_2 \sqsubseteq q_2 \Rightarrow p_1 * p_2 \sqsubseteq q_1 * q_2$$

*Atomic satisfaction* $\boxed{sat_{(-)} : \mathcal{P}(\Delta(RId)) \to \mathcal{P}(Action \times Prop \times Prop) \in \mathcal{S}}$

$$\begin{aligned} a \text{ sat}_A \ \{p\}\,\{q\} \stackrel{\text{def}}{=} \forall r \in Prop.\ \forall m \in \mathcal{M}.\ \forall h, h' \in Heap. \\ m \in p * \triangleright r \wedge h \in \lfloor m \rfloor_A \wedge h' \in [\![a]\!](h) \wedge stable_A(r) \\ \Rightarrow\ \exists m' \in \mathcal{M}.\ \triangleright (m' \in q * r \wedge h' \in \lfloor m' \rfloor_A) \end{aligned}$$

We use $a$ sat $\{p\}\,\{q\}$ as shorthand for $a$ sat$_{RId}$ $\{p\}\,\{q\}$.

**Lemma 8.**

$$\begin{aligned} \forall A \in \mathcal{P}(\Delta(RId)).\ \forall p_1, p_2, q_1, q_2 \in Prop. \\ p_1 \sqsubseteq_A p_2 \wedge a\ sat_A\ \{p_2\}\,\{q_2\} \wedge \triangleright(q_2 \sqsubseteq_A q_1) \Rightarrow a\ sat_A\ \{p_1\}\,\{q_1\} \end{aligned}$$

*Proof.* Assume $r \in Prop$, $m \in \mathcal{M}$, and $h, h' \in Heap$ such that

$$stable_A(r) \qquad m \in p_1 * \triangleright r \qquad h \in \lfloor m \rfloor_A \qquad h' \in [\![a]\!](h)$$

By Lemma 5 it follows that $stable_A(\triangleright r)$ and thus

$$h \in \lfloor p * \triangleright r \rfloor_A \subseteq \lfloor q * \triangleright r \rfloor_A$$

Hence, there exists $m' \in \mathcal{M}$ such that

$$\triangleright(m' \in q_2 * r \wedge h' \in \lfloor m' \rfloor_A)$$

Then

$$\triangleright(m' \in \lfloor q_2 * r \rfloor_A \subseteq \lfloor q_1 * r \rfloor_A)$$

□

**Lemma 9.**

$$\begin{aligned} \forall A \in \mathcal{P}(\Delta(RId)).\ \forall p, q, r \in Prop. \\ stable_A(r) \wedge a\ sat_A\ \{p\}\,\{q\} \Rightarrow a\ sat_A\ \{p * \triangleright r\}\,\{q * r\} \end{aligned}$$

*Proof.* Assume $c \in Prop$, $m \in \mathcal{M}$, and $h, h' \in Heap$ such that

$$stable_A(c) \qquad m \in p * \triangleright r * \triangleright c \qquad h \in \lfloor m \rfloor_A \qquad h' \in [\![a]\!](h)$$

Then $m \in p * \triangleright(r * c)$. Since $stable_A(r * c)$ it follows that there exists an $m' \in \mathcal{M}$ such that

$$\triangleright(m' \in q * r * c \wedge h' \in \lfloor m' \rfloor_A)$$

□

**Corollary 1.**

$$\forall A \in \mathcal{P}(\Delta(RId)).\ \forall p, q, r \in Prop.$$
$$stable_A(r) \wedge a\ sat_A\ \{p\}\,\{q\} \Rightarrow a\ sat_A\ \{p * r\}\,\{q * r\}$$

**Lemma 10.**

$$\forall A \in \mathcal{P}(\Delta(RId)).\ \forall p \in Prop.\ id\ sat_A\ \{p\}\,\{p\}$$

*Proof.* Assume $r \in Prop$, $m \in \mathcal{M}$ and $h, h' \in Heap$ such that

$$stable_A(r) \qquad m \in p * \triangleright r \qquad h \in \lfloor m \rfloor_A \qquad h' \in [\![id]\!](h)$$

Then $h' = h$ and there exists $m_1, m_2 \in \mathcal{M}$ such that

$$m = m_1 \bullet m_2 \qquad m_1 \in p \qquad m_2 \in \triangleright r$$

Thus, $m_1 \in \triangleright p$ from which it follows that $\triangleright(m \in p * r)$. Thus,

$$\triangleright(m \in p * r \wedge h' \in \lfloor m \rfloor_A)$$

□

*Specification embedding* $\boxed{spec : \Omega \to Prop \in \mathcal{S}}$

$$spec(s) \stackrel{\text{def}}{=} \{m \in \mathcal{M} \mid s\}$$

**Lemma 11.**

$$\forall a \in Action.\ \forall p, q \in Prop.\ \forall s \in \Omega.$$
$$a\ sat\ \{p * spec(\triangleright s)\}\{q\} \Rightarrow a\ sat\ \{p * spec(\triangleright s)\}\{q * spec(s)\}$$

*Proof.* Let $r \in Prop$, $m \in \mathcal{M}$ and $h, h' \in Heap$ such that

$$m \in p * spec(\triangleright s) * \triangleright r \qquad h \in \lfloor m \rfloor \qquad h' \in [\![a]\!](h) \qquad stable(r)$$

Hence, by assumption, there exists an $m' \in \mathcal{M}$ such that

$$\triangleright(m' \in q * r) \qquad \triangleright(h' \in \lfloor m' \rfloor)$$

Hence, $\triangleright((m' \in q * r) \wedge s)$ and thus $\triangleright(m' \in q * spec(s) * r)$. □

**Lemma 12.**

$$\triangleright(p * q) = (\triangleright p) * (\triangleright q)$$

*Proof ($\subseteq$).* Assume $m \in \triangleright(p * q)$ then $\triangleright(m \in p * q)$ and thus

$$\triangleright(\exists m_1, m_2 \in \mathcal{M}.\ m = m_1 \bullet m_2 \wedge m_1 \in p \wedge m_2 \in q)$$

Unfolding the definition of $\bullet$, it follows that

$$\begin{array}{l}(\exists m_1, m_2 \in \mathcal{M}.\ m = m_1 \bullet m_2 \wedge m_1 \in p \wedge m_2 \in q) \Rightarrow \\ \quad (\exists l_1, l_2 \in \Delta(\mathit{LState}).\ m.l = l_1 \bullet_{\mathit{LState}} l_2 \wedge (l_1, m.s, m.a) \in p \wedge (l_2, m.s, m.a) \in q)\end{array}$$

Hence there exists $l_1, l_2 \in \Delta(\mathit{LState})$ such that

$$\triangleright(m.l = l_1 \bullet_{\mathit{LState}} l_2) \qquad \triangleright((l_1, m.s, m.a) \in p) \qquad \triangleright((l_2, m.s, m.a) \in q)$$

By Lemma 1, it follows that $m.l = l_1 \bullet_{\mathit{LState}} l_2 \vee \triangleright\bot$.

Case $m.l = l_1 \bullet_{\mathit{LState}} l_2$: then $m = (l_1, m.s, m.a) \bullet (l_2, m.s, m.a)$ and thus

$$m \in (\triangleright p) * (\triangleright q)$$

Case $\triangleright\bot$: trivially, $m = m \bullet (\varepsilon, m.s, m.a)$ and since $\triangleright\bot$, $\triangleright(m \in p)$ and $\triangleright((\varepsilon, m.s, m.a) \in q)$ and thus,

$$m \in (\triangleright p) * (\triangleright q)$$

□

*Proof of ($\supseteq$).* Follows easily from monotonicity of $\triangleright$. □

*Thread safety* $\boxed{\mathit{safe} : \mathcal{P}(\mathit{Thread} \times \mathit{Prop} \times (\Delta(\mathit{Stack}) \to \mathit{Prop})) \in \mathcal{S}}$

Define *safe* by guarded recursion,

$$\mathit{safe} = \mathsf{fix}\ f.\ \widehat{\mathit{safe}}(f)$$

where

$$\begin{array}{rl}\widehat{\mathit{safe}}(f)(x,p,q) \stackrel{\text{def}}{=} & (\mathit{irr}(x) \wedge p \sqsubseteq q(x.l)) \vee \\ & (\forall T \in \mathit{TPool}.\ \forall a \in \mathit{Action}.\ \forall y \in \mathit{Thread}.\ x \xrightarrow{a} \{y\} \uplus T \wedge x.t = y.t\ \Rightarrow \\ & \quad \exists p' \in \{y\} \uplus T \to \mathit{Prop}. \\ & \qquad \forall z \in (\{y\} \uplus T).\ \triangleright\, \mathit{stable}(p'(z)) \wedge \\ & \qquad a\ \mathit{sat}\ \{p\}\{p'(y) * \circledast_{z \in T} p'(z)\} \wedge \\ & \qquad \triangleright\, f(y, p'(y), q) \wedge \\ & \qquad \triangleright\, \forall z \in T.\ f(z, p'(z), \lambda l'.\ \top))\end{array}$$

**Lemma 13.**

$$\begin{array}{l}\forall X \in \mathcal{P}(\Delta(\mathit{SId})).\ \forall p \in \Delta(\mathit{LTS}).\ \forall r \in \Delta(\mathit{RId}). \\ \quad (\forall \alpha \in \mathit{AId}.\ \alpha \neq \alpha_i \Rightarrow p(\alpha)(X) \subseteq X) \Rightarrow \mathit{stable}(\mathit{region}(X, p, r) * [\alpha_1]^r_1 * \cdots * [\alpha_n]^r_1)\end{array}$$

**Lemma 14.**

$$\forall A, B \in \mathcal{P}(\Delta(RId)).\ \forall p, q \in Prop.\ A \subseteq B \wedge p \sqsubseteq_A q \Rightarrow p \sqsubseteq_B q$$

**Lemma 15.**

$$\forall p, q, r \in Prop.\ \forall A \in \mathcal{P}(\Delta(RId)).\ p \sqsubseteq_A q \wedge q \sqsubseteq_A r \Rightarrow p \sqsubseteq_A r$$

**Lemma 16.**

$$\begin{aligned}&\forall p \in \Delta(SId) \to Prop.\ stable(p) \Rightarrow \\ &\quad (\lambda(x, \varsigma).\ \{(l, s) \mid (l, s, \varsigma) \in p(x)\}) \in \Delta(SId) \times AMod \to_{mon} \mathcal{P}^{\uparrow}(AState)\end{aligned}$$

*Proof (Monotonicity).* Assume $x \in \Delta(SId)$ and $\varsigma_1, \varsigma_2 \in AMod$ such that $\varsigma_1 \leq \varsigma_2$. By upwards-closure of $p$, for any $l \in LState$ and $s \in SState$ such that $(l, s, \varsigma_1) \in p(x)$ then $(l, s, \varsigma_2) \in p(x)$. □

*Proof (Upwards-closure).* Assume $x \in \Delta(SId)$, $\varsigma \in AMod$, $l_1, l_2 \in LState$ and $s_1, s_2 \in SState$ such that $(l_1, s_1, \varsigma) \in p(x)$ and $(l_1, s_1)\ R\ (l_2, s_2)$. By stability of $p$ it follows that $(l_2, s_2, \varsigma) \in p(x)$. □

**Lemma 17.**

$$\begin{aligned}&\forall U \in \mathcal{P}(\Delta(RId)).\ \forall T \in \Delta(LTS).\ \forall X \in \mathcal{P}(\Delta(SId)).\ \forall x \in X. \\ &\forall I \in \Delta(RId) \to \Delta(SId) \to Prop.\ \forall P \in Prop.\ \forall A, B \in \mathcal{P}(\Delta(AId)) \\ &\quad (\forall r \in \Delta(RId).\ \forall s \in \Delta(SId).\ stable(I(r)(s))) \wedge U\ \textit{is infinite} \wedge A\ \textit{and}\ B\ \textit{finite} \wedge \\ &\quad valid(\forall n \in U.\ P \circledast_{\alpha \in A} [\alpha]_1^n \Rightarrow \triangleright I(n)(x)) \wedge A \cap B = \emptyset \\ &\qquad \Rightarrow P \sqsubseteq_U \exists n \in U.\ region(X, T, n) * rintr(I(n), n) * \circledast_{\alpha \in B} [\alpha]_1^n\end{aligned}$$

*Proof.* Assume $r \in Prop$, $m \in \mathcal{M}$ and $h \in \Delta(Heap)$ such that

$$m \in P * r \qquad\qquad h \in \lfloor m \rfloor_A$$

Hence, there exists $l_1, l_2, l_c \in LState$, $s \in SState$, $\varsigma \in AMod$ and $sr : dom(s) \cap U \to LState$ such that

$$h = l_c.h \qquad l_c = l_1 \bullet l_2 \bullet \Pi_{r \in dom(s) \cap A} sr(r) \qquad \forall r \in dom(s) \cap U.\ sr(r) \in \lfloor (s, \varsigma) \rfloor_r$$

and

$$m = (l_1 \bullet l_2, s, \varsigma) \qquad (l_1, s, \varsigma) \in P \qquad (l_2, s, \varsigma) \in r$$

Pick an $n \in U$ such that $\forall \alpha \in AId.\ l_c.c(n, \alpha) = 0$ and $n \notin (dom(s) \cap dom(\varsigma))$. Let

$$s' = s[n \mapsto (x, T)] \qquad \varsigma' = \varsigma[n \mapsto lam(\lambda(y, \varsigma).\ \{(l, s) \in AState \mid (l, s, \varsigma) \in I(y)\})]$$

Note that $\varsigma'$ is well-defined thanks to Lemma 16. Then $s \leq s'$ and $\varsigma \leq \varsigma'$ and thus

$$(l_1, s', \varsigma') \in P \qquad\qquad (l_2, s', \varsigma') \in r$$

Furthermore, $((\varepsilon, [(n, A) \mapsto 1]), s', \varsigma') \in \circledast_{\alpha \in A} [\alpha]_1^n$ and thus

$$(l_1 \bullet (\varepsilon, [(n, A) \mapsto 1]), s', \varsigma') \in \triangleright I(n)(x)$$

as $P \circledast_{\alpha \in A} [\alpha]_1^n \Rightarrow \triangleright I(n)(x)$.

Clearly, $(\varepsilon, s', \varsigma') \in region(X, T, n)$. Furthermore, for every $\varsigma'' \geq \varsigma'$ and $y \in \Delta(SId)$,

$$\begin{aligned} app(\varsigma'(n))(y, \varsigma'') &= app(lam(\lambda(y, \varsigma).\ \{(l, s) \mid (l, s, \varsigma) \in I(y)\}))(y, \varsigma'') \\ &= (\triangleright(\lambda(y, \varsigma).\ \{(l, s) \mid (l, s, \varsigma) \in I(y)\}))(y, \varsigma'') \\ &= \{(l, s) \mid (l, s, \varsigma'') \in \triangleright I(y)\} \end{aligned}$$

Thus, $(\varepsilon, s', \varsigma') \in rintr(I, n)$. Hence,

$$(l_2 \bullet (\varepsilon, [(n, B) \mapsto 1]), s', \varsigma') \in region(X, T, n) * rintr(I, n) * (\circledast_{\alpha \in B} [\alpha]_1^n) * r$$

Lastly,

$$h \in \lfloor (l_2 \bullet (\varepsilon, [(n, B) \mapsto 1]), s', \varsigma') \rfloor_U$$

□

**Lemma 18.**

$\forall A \in \mathcal{P}(\Delta(RId)).\ \forall X, Y \in \mathcal{P}(\Delta(SId)).\ \forall T \in \Delta(LTS).\ \forall n \in \Delta(RId).$
$\forall I \in \Delta(SId) \to Prop.\ \forall p, q \in Prop.\ \forall \alpha \in \Delta(AId).\ \forall \pi \in \Delta(Perm_n).\ \forall f : X \to Y.$
$stable(p) \wedge n \in A \wedge (\forall x \in X.\ (x, f(x)) \in T(\alpha) \vee f(x) = x) \wedge$
$(\forall x \in X.\ p * (\triangleright I(x)) * [\alpha]_\pi^n \sqsubseteq_{A \setminus \{n\}} q * \triangleright I(f(x)))$
$\Rightarrow (region(X, T, n) * rintr(I, n) * p * [\alpha]_\pi^n \sqsubseteq_A region(Y, T, n) * q)$

*Proof.* Assume $r \in Prop$, $m \in \mathcal{M}$ and $h \in \Delta(Heap)$ such that

$$m \in region(X, T, n) * rintr(I, n) * p * [\alpha]_\pi^n * r \qquad h \in \lfloor m \rfloor_A \qquad stable_A(r)$$

hence there exists $l_1, l_2, l_3, l_4, l_5 \in LState$, $s \in SState$ and $\varsigma \in AMod$ such that

$$(l_1, s, \varsigma) \in region(X, T, n) \qquad (l_2, s, \varsigma) \in rintr(I, n) \qquad (l_3, s, \varsigma) \in p$$

and

$$(l_4, s, \varsigma) \in [\alpha]_\pi^n \qquad (l_5, s, \varsigma) \in r \qquad m = (l_1 \bullet l_2 \bullet l_3 \bullet l_4 \bullet l_5, s, \varsigma)$$

Let $l = l_1 \bullet l_2 \bullet l_3 \bullet l_4 \bullet l_5$. Unfolding $h \in \lfloor m \rfloor_A$ there thus exists an $l_c \in \mathit{LState}$ and $sr : (\mathit{dom}(s) \cap A) \to \mathit{LState}$ such that

$$h = l_c.h \qquad l_c = l \bullet \Pi_{r \in (\mathit{dom}(s) \cap A)} sr(r) \qquad \forall r \in (\mathit{dom}(s) \cap A).\ sr(r) \in \lfloor (s, \varsigma) \rfloor_r$$

Since $(l_2, s, \varsigma) \in \mathit{region}(X, T, n)$ it follows that $n \in \mathit{dom}(s)$, $s(n).s \in X$ and $s(n).p = T$. Since $n \in \mathit{dom}(s)$ and $n \in A$ it follows that $sr(n) \in \lfloor (s, \varsigma) \rfloor_n$ and thus $(sr(n), s) \in \mathit{app}(\varsigma(n))(s(n).s, \varsigma)$. Unfolding $(l_1, s, \varsigma) \in \mathit{rintr}(I, n)$ it follows that,

$$\forall x \in \Delta(\mathit{SId}).\ \forall \varsigma' \geq \varsigma.\ \mathit{app}(\varsigma(n))(x, \varsigma') = \triangleright(\lambda(l, s).\ I(x)(l, s, \varsigma'))$$

Hence in particular, $\mathit{app}(\varsigma(n))(s(n).s, \varsigma) = \triangleright(\lambda(l, s).\ I(s(n).s)(l, s, \varsigma)$ and thus,

$$\triangleright I(s(n).s)(sr(n), s, \varsigma)$$

By assumption, $(s(n).s, f(s(n).s)) \in T(\alpha)$. Furthermore, since $(l_4, s, \varsigma) \in [\alpha]^n_\pi$ it follows that $\pi \leq l_4.c(n, \alpha)$ and thus $l_3.c(n, \alpha) < 1$, $sr(n).c(n, \alpha) < 1$ and $l_5.c(n, \alpha) < 1$. Hence,

$$(l_3, s)\ R_A\ (l_3, s') \qquad (l_5, s)\ R_A\ (l_5, s') \qquad (sr(n), s)\ R_A\ (sr(n), s')$$

where $s' = s[n \mapsto (f(s(n).s), s(n).p)]$. Hence, by stability of $p$, $\triangleright I(s(n).s)$ and $r$, it follows that,

$$(l_3, s', \varsigma) \in p \qquad (sr(n), s', \varsigma) \in \triangleright I(s(n).s) \qquad (l_4, s', \varsigma) \in [\alpha]^n_\pi \qquad (l_5, s', \varsigma) \in r$$

Furthermore, for every $r \in \mathit{dom}(s) \cap (A \setminus \{n\})$, $sr(r).c(n, \alpha) < 1$ and thus

$$(sr(r), s)\ R_A\ (sr(r), s')$$

Hence, by stability of the region interpretations,

$$\forall r \in \mathit{dom}(s') \cap (A \setminus \{n\}).\ sr'(r) \in \lfloor (s', \varsigma) \rfloor_r$$

where $sr'$ is defined as,

$$sr'(r) \stackrel{\text{def}}{=} \begin{cases} sr(r) & \text{if } r \neq n \\ l'_2 & \text{if } r = n \end{cases}$$

We thus have that

$$\begin{array}{l} (l_1 \bullet l_2 \bullet l_3 \bullet sr(n) \bullet l_4 \bullet l_5, s', \varsigma) \in \\ \quad p * (\triangleright I(s(n).s)) * [\alpha]^n_\pi * r * \mathit{region}(\{f(s(n).s)\}, T, n) * \{m \in \mathcal{M} \mid \varsigma \leq m.a\} \end{array}$$

and

$$h \in \lfloor (l_1 \bullet l_2 \bullet l_3 \bullet sr(n) \bullet l_4 \bullet l_5, s', \varsigma) \rfloor_{A \setminus \{n\}}$$

Hence, from the assumed view-shift it follows that there exists an $m' \in \mathcal{M}$ such that

$$m' \in q * (\triangleright I(f(s(n).s))) * r * region(\{f(s(n).s)\}, T, n) * \{m \in \mathcal{M} \mid \varsigma \leq m.a\}$$

and $h \in \lfloor m' \rfloor_{A \setminus \{n\}}$. There thus exists $l'_1, l'_2, l'_3, l'_4, l'_5 \in \mathit{LState}$, $s'' \in \mathit{SState}$, and $\varsigma' \in \mathit{AMod}$ such that

$$m' = (l'_1 \bullet l'_2 \bullet l'_3 \bullet l'_4 \bullet l'_5, s'', \varsigma') \qquad (l'_1, s'', \varsigma') \in q \qquad (l'_2, s'', \varsigma') \in \triangleright I(f(s(n).s))$$

and

$$(l'_3, s'', \varsigma') \in r \quad (l'_4, s'', \varsigma') \in region(\{f(s(n).s)\}, T, n) \quad (l'_5, s'', \varsigma') \in \{m \in \mathcal{M} \mid \varsigma \leq m.a\}$$

Hence, $s''(n).s = f(s(n).s) = s'(n).s$. From $h \in \lfloor m' \rfloor_{A \setminus \{n\}}$ it follows that there exists an $l'_c \in \mathit{LState}$ and $sr'' : dom(s'') \cap (A \setminus \{n\}) \to \mathit{LState}$ such that

$$h = l'_c.h \qquad\qquad l'_c = l'_1 \bullet l'_2 \bullet l'_3 \bullet l'_4 \bullet l'_5 \bullet \Pi_{r \in dom(s'') \cap (A \setminus \{n\})} sr''(r)$$

and $\forall r \in dom(s'') \cap (A \setminus \{n\}).\ sr''(r) \in \lfloor (s'', \varsigma') \rfloor_r$. From $(l'_5, s'', \varsigma') \in \{m \in \mathcal{M} \mid \varsigma \leq m.a\}$ it follows that $\varsigma \leq \varsigma'$ and thus,

$$\forall x \in \Delta(\mathit{SId}).\ app(\varsigma'(n))(x, \varsigma') = \triangleright(\lambda(l, s).\ I(x)(l, s, \varsigma'))$$

Hence, $(l'_2, s'') \in app(\varsigma'(n))(f(s(n).s), \varsigma')$ and thus $l'_2 \in \lfloor (s'', \varsigma') \rfloor_n$. Thus,

$$h \in \lfloor region(Y, T, n) * q \rfloor_A$$

□

**Lemma 19.**

$\forall A \in \mathcal{P}(\Delta(\mathit{RId}))$. $\forall X, Y \in \mathcal{P}(\Delta(\mathit{SId}))$. $\forall T \in \Delta(\mathit{LTS})$. $\forall n \in \Delta(\mathit{RId})$. $\forall \pi \in \Delta(\mathit{Perm}_n)$.
$\forall I \in \Delta(\mathit{SId}) \to \mathit{Prop}$. $\forall p, p' \in \mathit{Prop}$. $\forall q \in Y \to \mathit{Prop}$. $\forall \alpha \in \Delta(\mathit{AId})$.
$\forall f : X \to Y$. $\forall g : \Delta(\mathit{AId}) \to \Delta(\mathsf{Perm})$.

$$\begin{array}{l} \mathit{stable}(p) \wedge n \in A \wedge (\forall x \in X.\ (x, f(x)) \in \overline{T(B)} \vee f(x) = x) \ \wedge \\ (\forall x \in X.\ a\ \mathit{sat}_{A \setminus \{n\}}\ \left\{p * \circledast_{\alpha \in B} [\alpha]^n_{g(\alpha)} * (\triangleright I(x))\right\} \{q(x) * \triangleright I(f(x)))\} \\ \quad \Rightarrow a\ \mathit{sat}_A\ \left\{region(X, T, I, n) * p * \circledast_{\alpha \in B} [\alpha]^n_{g(\alpha)}\right\} \{\exists y \in Y.\ region(\{f(y)\}, T, n) * q(y)\} \end{array}$$

*Proof.* Assume $r \in \mathit{Prop}$, $m \in \mathcal{M}$ and $h, h' \in \Delta(\mathit{Heap})$ such that

$$m \in region(X, T, I, n) * p * \triangleright r \qquad h \in \lfloor m \rfloor_A \qquad h' \in [\![a]\!](h) \qquad \mathit{stable}_A(r)$$

hence there exists $l_1, l_2, l_3, l_4, l_5 \in \mathit{LState}$, $s \in \mathit{SState}$ and $\varsigma \in \mathit{AMod}$ such that

$$(l_1, s, \varsigma) \in region(X, T, n) \quad (l_2, s, \varsigma) \in rintr(I, n) \quad (l_3, s, \varsigma) \in p \quad (l_4, s, \varsigma) \in \circledast_{\alpha \in B} [\alpha]^n_{g(\alpha)}$$

and

$$(l_5, s, \varsigma) \in \triangleright r \qquad\qquad m = (l_1 \bullet l_2 \bullet l_3 \bullet l_4 \bullet l_5, s, \varsigma)$$

Let $l = l_1 \bullet l_2 \bullet l_3 \bullet l_4 \bullet l_5$. Unfolding $h \in \lfloor m \rfloor_A$ there thus exists an $l_c \in \mathit{LState}$ and $sr : (\mathit{dom}(s) \cap A) \to \mathit{LState}$ such that

$$h = l_c.h \qquad l_c = l \bullet \Pi_{r \in (\mathit{dom}(s) \cap A)} sr(r) \qquad \forall r \in (\mathit{dom}(s) \cap A).\ sr(r) \in \lfloor (s, \varsigma) \rfloor_r$$

Since $(l_2, s, \varsigma) \in \mathit{region}(X, T, n)$ it follows that $n \in \mathit{dom}(s)$, $s(n).s \in X$ and $s(n).p = T$. Since $n \in \mathit{dom}(s)$ and $n \in A$ it follows that $sr(n) \in \lfloor (s, \varsigma) \rfloor_n$ and thus $(sr(n), s) \in \mathit{app}(\varsigma(n))(s(n).s, \varsigma)$. Unfolding $(l_1, s, \varsigma) \in \mathit{rintr}(I, n)$ it follows that,

$$\forall x \in \Delta(\mathit{SId}).\ \forall \varsigma' \geq \varsigma.\ \mathit{app}(\varsigma(n))(x, \varsigma') = \triangleright(\lambda(l, s).\ I(x)(l, s, \varsigma'))$$

Hence in particular, $\mathit{app}(\varsigma(n))(s(n).s, \varsigma) = \triangleright(\lambda(l, s).\ I(s(n).s)(l, s, \varsigma)$ and thus,

$$\triangleright I(s(n).s)(sr(n), s, \varsigma)$$

Let $s' = s[n \mapsto (f(s(n).s), s(n).p)]$. By assumption, $(s(n).s, f(s(n).s)) \in \overline{T(B)} \vee f(s(n).s) = s(n).s$. If $f(s(n).s) = s(n).s$ then $s = s'$. Otherwise, since

$$(l_4, s, \varsigma) \in \circledast_{\alpha \in B} [\alpha]^n_{g(\alpha)}$$

it follows that

$$\forall l \in \{sr(n), l_3, l_5\}.\ B \subseteq \mathit{act}_{\mathit{allowed}}(l, n)$$

Hence,

$$\forall l \in \{sr(n), l_3, l_5\}.\ (s(n).s, f(s(n).s)) \in \overline{\mathit{upd}_{\mathit{allowed}}(l, n, T)}$$

and thus

$$(l_3, s)\ R_A\ (l_3, s') \qquad (l_5, s)\ R_A\ (l_5, s') \qquad (sr(n), s)\ R_A\ (sr(n), s')$$

Hence, by stability of $p$, $\triangleright I(s(n).s)$ and $r$, it follows that,

$$(l_3, s', \varsigma) \in p \qquad (sr(n), s', \varsigma) \in \triangleright I(s(n).s) \qquad (l_5, s', \varsigma) \in \triangleright r$$

Furthermore, for every $r \in \mathit{dom}(s) \cap (A \setminus \{n\})$, $sr(r).c(n, \alpha) < 1$ and thus

$$(sr(r), s)\ R_A\ (sr(r), s')$$

Hence, by stability of the region interpretations,

$$\forall r \in \mathit{dom}(s') \cap (A \setminus \{n\}).\ sr'(r) \in \lfloor (s', \varsigma) \rfloor_r$$

where $sr'$ is defined as,

$$sr'(r) \stackrel{\text{def}}{=} \begin{cases} sr(r) & \text{if } r \neq n \\ \text{undef} & \text{if } r = n \end{cases}$$

We thus have that

$$(l_1 \bullet l_2 \bullet l_3 \bullet sr(n) \bullet l_4 \bullet l_5, s', \varsigma) \in$$
$$p * (\triangleright I(s(n).s)) * (\circledast_{\alpha \in B} [\alpha]^n_{g(\alpha)}) * \triangleright r * region(\{f(s(n).s)\}, T, n) * \{m \in \mathcal{M} \mid \varsigma \leq m.a\}$$

and

$$h \in \lfloor (l_1 \bullet l_2 \bullet l_3 \bullet sr(n) \bullet l_4 \bullet l_5, s', \varsigma) \rfloor_{A \setminus \{n\}}$$

Hence, from the assumed view-shift it follows that there exists an $m' \in \mathcal{M}$ such that

$$\triangleright (m' \in q(f(s(n).s) * (\triangleright I(f(s(n).s))) * r$$
$$* \ region(\{f(s(n).s)\}, T, n) * \{m \in \mathcal{M} \mid \varsigma \leq m.a\})$$

and $\triangleright (h' \in \lfloor m' \rfloor_{A \setminus \{n\}})$.

There thus exists $l'_1, l'_2, l'_3, l'_4, l'_5 \in \mathit{LState}$, $s'' \in \mathit{SState}$, and $\varsigma' \in \mathit{AMod}$ such that

$$\triangleright m' = (l'_1 \bullet l'_2 \bullet l'_3 \bullet l'_4 \bullet l'_5, s'', \varsigma') \quad \triangleright (l'_1, s'', \varsigma') \in q(s(n).s) \quad \triangleright (l'_2, s'', \varsigma') \in \triangleright I(f(s(n).s))$$

and

$$\triangleright (l'_3, s'', \varsigma') \in r \quad \triangleright (l'_4, s'', \varsigma') \in region(\{f(s(n).s)\}, T, n) \quad \triangleright (l'_5, s'', \varsigma') \in \{m \in \mathcal{M} \mid \varsigma \leq m.a\}$$

Hence, $\triangleright (s''(n).s = f(s(n).s) = s'(n).s)$. From $\triangleright h \in \lfloor m' \rfloor_{A \setminus \{n\}}$ it follows that there exists an $l'_c \in \mathit{LState}$ and $sr'' : dom(s'') \cap (A \setminus \{n\}) \to \mathit{LState}$ such that

$$\triangleright h = l'_c.h \qquad \triangleright l'_c = l'_1 \bullet l'_2 \bullet l'_3 \bullet l'_4 \bullet l'_5 \bullet \Pi_{r \in dom(s'') \cap (A \setminus \{n\})} sr''(r)$$

and $\triangleright \forall r \in dom(s'') \cap (A \setminus \{n\})$. $sr''(r) \in \lfloor (s'', \varsigma') \rfloor_r$. From $\triangleright (l'_5, s'', \varsigma') \in \{m \in \mathcal{M} \mid \varsigma \leq m.a\}$ it follows that $\triangleright \varsigma \leq \varsigma'$ and thus,

$$\forall x \in \Delta(\mathit{SId}).\ app(\varsigma'(n))(x, \varsigma') = \triangleright (\lambda (l, s).\ I(x)(l, s, \varsigma'))$$

Hence, $\triangleright (l'_2, s'') \in app(\varsigma'(n))(f(s(n).s), \varsigma')$ and thus $\triangleright l'_2 \in \lfloor (s'', \varsigma') \rfloor_n$. Thus,

$$\triangleright h \in \lfloor region(\{f(s(n).s)\}, T, n) * q(s(n).s) \rfloor_A$$

□

**Definition 1.**

$$\overline{p} \stackrel{def}{=} p \vee \triangleright \bot \qquad p \in \Omega$$
$$\overline{p} \stackrel{def}{=} \{m \in \mathcal{M} \mid \overline{m \in p}\} \qquad p \in \mathit{Prop}$$

**Definition 2.**

$$x.f \mapsto y \stackrel{def}{=} \{(l, s, \varsigma) \in \mathcal{M} \mid l.h(x, f) = \mathit{Some}\ y\} \qquad x, y \in \Delta(\mathit{Val}), f \in \Delta(\mathit{FName})$$

**Lemma 20.**

$$\begin{array}{c} \forall x, y \in \Delta(\mathit{Val}).\ \forall f \in \Delta(\mathit{FName}).\ \forall m \in \mathcal{M}. \\ \triangleright\,(m \in x.f \mapsto y) \Rightarrow (m \in x.f \mapsto y) \vee \triangleright\bot \end{array}$$

*Proof.* We prove this externally, using the Kripke-Joyal forcing semantics. We thus have to prove that

$$\begin{array}{l} \forall n \in \mathsf{N}.\ n \models \forall x, y \in \Delta(\mathit{Val}).\ \forall f \in \Delta(\mathit{FName}).\ \forall m \in \mathcal{M}. \\ \qquad \triangleright\,(m \in x.f \mapsto y) \Rightarrow (m \in x.f \mapsto y) \vee \triangleright\bot \end{array}$$

By definition of the forcing semantics and some simplification, this reduces to,

$$\begin{array}{l} \forall n \in \mathsf{N}.\ \forall x, y \in \mathit{Val}.\ \forall f \in \mathit{FName}. \\ \quad n \models \forall m \in \mathcal{M}.\ \triangleright\,(m \in x.f \mapsto y) \Rightarrow (m \in x.f \mapsto y) \vee \triangleright\bot \end{array}$$

Unfolding further, this reduces to,

$$\begin{array}{l} \forall n \in \mathsf{N}.\ \forall x, y \in \mathit{Val}.\ \forall f \in \mathit{FName}.\ \forall k \leq n.\ \forall m \in (\Delta(\mathit{AState}) \times \mathit{AMod})(k).\ \forall j \leq k. \\ \quad (j \models \triangleright(m|_j \in x.f \mapsto y)) \Rightarrow ((j \models (m|_j \in x.f \mapsto y)) \vee (j \models \triangleright\bot)) \end{array}$$

Case $j = 1$: Then $j \models \triangleright\bot$.
Case $j > 1$: Then $j - 1 \models m|_{j-1} \in x.f \mapsto y$ and thus

$$j - 1 \models m|_{j-1}.l.h(x, f) = \mathit{Some}\ y$$

and thus,

$$j \models m|_j.l.h(x, f) = \mathit{Some}\ y$$

□

**Lemma 21.**

$$(\triangleright x.f \mapsto v) \Rightarrow \overline{x.f \mapsto v}$$

**Lemma 22.**

$$\begin{array}{c} \forall A \in \mathcal{P}(\Delta(\mathit{AId})).\ \forall p_1, p_2, q \in \mathit{Prop}.\ \forall a \in \Delta(\mathit{Action}). \\ p_1 \Rightarrow \overline{p_2} \wedge a\ \mathit{sat}_A\ \{p_2\}\,\{q\} \Rightarrow a\ \mathit{sat}_A\ \{p_1\}\,\{q\} \end{array}$$

*Proof.* Let $r \in \mathit{Prop}$, $m \in \mathcal{M}$ and $h, h' \in \mathit{Heap}$ such that

$$m \in p_1 * \triangleright r \qquad h \in \lfloor m \rfloor_A \qquad h' \in [\![a]\!](h) \qquad \mathit{stable}_A(r)$$

Then there exists $m_1, m_2 \in \mathcal{M}$ such that

$$m = m_1 \bullet m_2 \qquad m_1 \in p_1 \qquad m_2 \in \triangleright r$$

Hence, $m_1 \in \overline{p_2} = \{m \in \mathcal{M} \mid m \in p_2 \vee \triangleright \bot\}$.

Case $m_1 \in p_2$: Then $m \in p_2 * \triangleright r$ and there exists an $m' \in \mathcal{M}$ such that

$$\triangleright(m' \in q * r \wedge h' \in \lfloor m' \rfloor_A)$$

Case $\triangleright\bot$: Then $\triangleright(m' \in q * r \wedge h' \in \lfloor m' \rfloor_A)$ for any $m' \in \mathcal{M}$. □

*Thread-pool evaluation* $\boxed{\mathit{eval} : \Delta(\mathit{Heap}) \times \Delta(\mathit{TPool}) \times \mathcal{P}(\Delta(\mathit{Heap}) \times \Delta(\mathit{TPool})) \to \Omega}$

$$\mathit{eval}(h, T, q) \stackrel{\text{def}}{=} (\mathit{irr}(T) \wedge (h, T) \in q) \vee (\forall T', h'.\ (h, T) \to (h', T') \Rightarrow \triangleright \mathit{eval}(h', t', q))$$

**Lemma 23.**

$$\begin{array}{l}\forall T \in \mathcal{P}(\Delta(\mathit{TPool})).\ \forall p \in T \to \mathit{Prop}.\ \forall q \in T \to \Delta(\mathit{Stack}) \to \mathit{Prop}.\ \forall h \in \Delta(\mathit{Heap}).\\ \quad (\forall x \in T.\ \mathit{stable}(p(x))) \wedge (\forall x \in T.\ \forall s \in \Delta(\mathit{Stack}).\ \mathit{stable}(q(x)(s))) \wedge\\ \quad (\forall x \in T.\ \mathit{safe}(x, p(x), q(x))) \wedge h \in \lfloor \circledast_{x \in T} p(x) \rfloor \ \Rightarrow\\ \qquad \mathit{eval}(h, T, \lambda(h', T').\ h' \in \lfloor \circledast_{x \in T} q(x)(T'(x.t)) \rfloor)\end{array}$$

*where $T(t)$ is shorthand for the stack component of the thread in $T \in \mathcal{P}(\mathit{TPool})$ with thread identifier $t \in \mathit{TId}$.*

*Proof.* By Löb induction.

Case $\mathit{irr}(h, T)$: By assumption, $\forall x \in T.\ \mathit{irr}(x)$ and thus

$$\forall x \in T.\ p(x) \sqsubseteq q(x)(x.l)$$

Hence, by the frame-rule, $(\circledast_{x \in T} p(x)) \sqsubseteq (\circledast_{x \in T} q(x)(x.l))$ and thus

$$h \in \lfloor \circledast_{x \in T} q(x)(x.l) \rfloor$$

and $T(x.t) = x.l$ for $x \in T$.

Case $(h, T) \to (h', T')$: Hence, there exists $a \in \mathit{Action}$, $x \in T$ and $T'' \in \mathit{TPool}$ s.t.,

$$x \xrightarrow{a} T'' \qquad T' = (T \setminus \{x\}) \cup T'' \qquad h' \in [\![a]\!](h)$$

Hence, there exists a $y \in \mathit{Thread}$, $T''' \in \mathit{TPool}$, and $p' \in T'' \to \mathit{Prop}$ such that

$$x.t = y.t \qquad\qquad T'' = \{y\} \uplus T'''$$

and

$$\begin{array}{ll} \forall z \in T''.\ \triangleright\, \mathit{stable}(p'(z)) & a \text{ sat } \{p(x)\}\left\{\circledast_{z \in T''} p'(z)\right\} \\ \quad \triangleright \mathit{safe}(y, p'(y), q(x)) & \forall z \in T'''.\ \triangleright\, \mathit{safe}(z, p'(z), \lambda l'.\ \top) \end{array}$$

Let $p'' : T' \to \mathit{Prop}$ and $q'' : T' \to \Delta(\mathit{Stack}) \to \mathit{Prop}$ denote the following functions,

$$p''(z) = \begin{cases} p'(z) & \text{if } z \in T'' \\ p(z) & \text{if } z \in (T \setminus \{x\}) \end{cases} \qquad q''(z) = \begin{cases} q(z) & \text{if } z \in (T \setminus \{x\}) \\ q(x) & \text{if } z = y \\ \lambda l.\ \top & \text{if } z \in T''' \end{cases}$$

Hence, by the frame-rule it follows that

$$a \text{ sat } \{\circledast_{z \in T} p(z)\}\left\{\circledast_{z \in T'} p''(z)\right\}$$

and thus,

$$\triangleright(h' \in \lfloor \circledast_{z \in T'} p''(z) \rfloor)$$

Furthermore, by assumption and by monotonicity of $\triangleright$, it follows that

$$\forall z \in T'.\ \triangleright\, \mathit{stable}(p''(z))$$

and

$$\forall z \in T'.\ \triangleright\, \mathit{safe}(z, p''(z), q''(z))$$

Hence, by the induction hypothesis, $\triangleright \mathit{eval}(h', T', \lambda(h'', T'').\ h'' \in \lfloor \circledast_{z \in T'} q''(z)(T''(z.t)) \rfloor$. Lastly,

$$\forall T''.\ \circledast_{z \in T'} q''(z)(T''(z.t)) = (\circledast_{z \in T} q(z)(T''(z.t))) * (\circledast_{z \in T'''} \top) = \circledast_{z \in T} q(z)(T''(z.t))$$

□

## Nested specifications

**Lemma 24.**

$$\begin{array}{l} \forall x \in \Delta(\mathit{Thread}).\ \forall p \in \mathit{Prop}.\ \forall q \in \Delta(\mathit{Stack}) \to \mathit{Prop}. \\ \quad \mathit{safe}(x, p * \mathit{spec}(\bot), q) \end{array}$$

**Lemma 25.**

$$\begin{array}{l} \forall s \in \Omega.\ \forall x \in \Delta(\mathit{Thread}).\ \forall p \in \mathit{Prop}.\ \forall q \in \Delta(\mathit{Stack}) \to \mathit{Prop}. \\ \quad (s \Rightarrow \mathit{safe}(x, p, q)) \Rightarrow \mathit{safe}(x, p * \mathit{spec}(s), q) \end{array}$$

*Proof.*

- By Corollary 2 it suffices to prove $\phi(\bot)$ and

$$\triangleright(\forall s : \Omega.\ \phi(s)) \Rightarrow \forall s : \Omega.\ \phi(\triangleright s)$$

  where

$$\begin{aligned}\phi \stackrel{\text{def}}{=} \lambda s : \Omega.\ \forall x \in \Delta(\mathit{Thread}).\ \forall p \in \mathit{Prop}.\ \forall q \in \Delta(\mathit{Stack}) \to \mathit{Prop}.\\ (s \Rightarrow \mathit{safe}(x, p, q)) \Rightarrow \mathit{safe}(x, p * \mathit{spec}(s), q))\end{aligned}$$

- $\phi(\bot)$ follows from Lemma 24
- thus, assume

$$\begin{aligned}\triangleright\,(\forall s : \Omega.\ \forall x \in \Delta(\mathit{Thread}).\ \forall p \in \mathit{Prop}.\ \forall q \in \Delta(\mathit{Stack}) \to \mathit{Prop}.\\ (s \Rightarrow \mathit{safe}(x, p, q)) \Rightarrow \mathit{safe}(x, p * \mathit{spec}(s), q))\end{aligned}$$

  and let $s \in \Omega$, $x \in \Delta(\mathit{Thread})$, $p \in \mathit{Prop}$ and $q : \Delta(\mathit{Stack}) \to \mathit{Prop}$ such that

$$\triangleright s \Rightarrow \mathit{safe}(x, p, q)$$

- case $\mathit{irr}(x)$ :
  - then $\triangleright s \Rightarrow p \sqsubseteq q(x_l)$
  - thus (by Lemma 27)

$$p * \mathit{spec}(\triangleright s) \sqsubseteq q(x_l)$$

  - hence

$$\mathit{safe}(x, p * \mathit{spec}(\triangleright s), q)$$

- case $x \xrightarrow{a} y \uplus T$ with $x_t = y_t$:
  - then

$$\begin{aligned}\triangleright s \Rightarrow (\ &\exists p' : y \uplus T \to \mathit{Prop}.\\ &a\ \mathit{sat}\ \{p\}\{\circledast_{z \in (y \uplus T)} p'(z)\} \wedge\\ &\triangleright\, \mathit{safe}(y, p'(y), q) \wedge\\ &\triangleright\, \forall z \in T.\ \mathit{safe}(z, p'(z), \lambda\_.\ \top))\end{aligned}$$

  - and thus (as $y \uplus T \to \mathit{Prop}$ is total and inhabited)

$$\begin{aligned}\exists p' : y \uplus T \to \mathit{Prop}.\ \triangleright\, s \Rightarrow (\ &a\ \mathit{sat}\ \{p\}\{\circledast_{z \in (y \uplus T)} p'(z)\} \wedge\\ &\triangleright\, \mathit{safe}(y, p'(y), q) \wedge\\ &\triangleright\, \forall z \in T.\ \mathit{safe}(z, p'(z), \lambda\_.\ \top))\end{aligned}$$

– let $p''(z) = p'(z) * spec(s)$

– then

$$\begin{aligned}&\triangleright s \Rightarrow a \ sat \ \{p\}\{\circledast_{z\in(y\uplus T)} p'(z)\}\\ &\triangleright s \Rightarrow \triangleright safe(y, p'(y), q)\\ &\triangleright s \Rightarrow \forall z \in T.\ \triangleright\ safe(z, p'(z), \lambda\_.\ \top)\end{aligned}$$

– and thus (by Lemma 28)

$$a \ sat \ \{p * spec(\triangleright s)\}\{\circledast_{z\in(y\uplus T)} p'(z)\}$$

– hence (by Lemma 11)

$$a \ sat \ \{p * spec(\triangleright s)\}\{(\circledast_{z\in(y\uplus T)} p'(z)) * spec(s)\}$$

– and since

$$\circledast_{z\in(y\uplus T)} p''(z) = (\circledast_{z\in(y\uplus T)} p'(z)) * spec(s)$$

– it follows that

$$a \ sat \ \{p * spec(\triangleright s)\}\{\circledast_{z\in(y\uplus T)} p''(z)\}$$

□

**Lemma 26.**

$$\begin{aligned}&\forall x \in \Delta(\mathit{Thread}).\ \forall p \in \mathit{Prop}.\ \forall q \in \Delta(\mathit{Stack}) \to \mathit{Prop}.\ \forall s \in \Omega.\\ &\quad safe(x, p * spec(s), q) \Rightarrow (s \Rightarrow safe(x, p, q))\end{aligned}$$

*Proof.*

- assume $safe(x, p * spec(s), q)$ and $s$
- then $p \sqsubseteq p * spec(s)$
- an thus, by the rule-of-consequence, $safe(x, p, q)$

□

**Lemma 27.**

$$\begin{aligned}&\forall p, q \in \mathit{Prop}.\ \forall s \in \Omega.\\ &\qquad (s \Rightarrow p \sqsubseteq q) \Rightarrow (p * spec(s) \sqsubseteq q)\end{aligned}$$

*Proof.*

- assume $r \in \mathit{Prop}$ such that

$$s \Rightarrow p \sqsubseteq q \qquad m \in p * \mathit{spec}(s) * r \qquad h \in \lfloor m \rfloor$$

- then by assumption there exists $m_1, m_2, m_3$ such that

$$m_1 \in p \qquad m_2 \in \mathit{spec}(s) \qquad m_3 \in r \qquad m = m_1 \cdot m_2 \cdot m_3$$

- hence $s$ holds and $m_2$ is the monoid unit
- hence $p \sqsubseteq q$ and there thus exists an $m' \in q * r$ such that $h \in \lfloor m' \rfloor$

□

**Lemma 28.**

$$\forall a \in \Delta(\mathit{Action}).\ \forall p, q \in \mathit{Prop}.\ \forall s \in \Omega.$$
$$(s \Rightarrow a\ \mathit{sat}\ \{p\}\{q\}) \Rightarrow a\ \mathit{sat}\ \{p * \mathit{spec}(s)\}\{q\}$$

$$\lceil p \rceil = \{(l, s, \vartheta) \in \mathcal{M} \mid \exists l', s'.\ (l', s', \vartheta) \in p \wedge (l', s')\ R\ (l, s)\}$$
$$\lfloor p \rfloor = \{(l, s, \vartheta) \in \mathcal{M} \mid \forall l', s'.\ (l, s)\ R\ (l', s') \Rightarrow (l', s', \vartheta) \in p\}$$

**Lemma 29** ($\leq \subseteq R$)**.**

$$\forall l_1, l_2 \in \Delta(\mathit{LState}).\ \forall s_1, s_2 \in \Delta(\mathit{SState}).\ l_1 \leq l_2 \wedge s_1 \leq s_2 \Rightarrow (l_1, s_1)\ R\ (l_2, s_2)$$

**Lemma 30** ($\lceil - \rceil$ is well-defined.)**.**

$$\forall p \in \mathit{Prop}.\ \forall l_1, l_2 \in \Delta(\mathit{LState}).\ \forall s_1, s_2 \in \Delta(\mathit{SState}).\ \forall \vartheta_1, \vartheta_2 \in \mathit{AMod}.$$
$$(l_1, s_1, \vartheta_1) \leq (l_2, s_2, \vartheta_2) \wedge (l_1, s_1, \vartheta_1) \in \lceil p \rceil \Rightarrow (l_2, s_2, \vartheta_2) \in \lceil p \rceil$$

*Proof.* By assumption there exists $l_1', s_1'$ such that

$$(l_1', s_1', \vartheta_1) \in p \qquad\qquad (l_1', s_1')\ R\ (l_1, s_1)$$

Since $p$ is upwards-closed it follows that $(l_1', s_1', \vartheta_2) \in p$. Furthermore, by Lemma 29 it follows that $(l_1, s_1)\ R\ (l_2, s_2)$. Thus, by transitivity, $(l_1', s_1')\ R\ (l_2, s_2)$ and thus

$$(l_2, s_2, \vartheta_2) \in \lceil p \rceil.$$

□

**Lemma 31.**

$$\forall p, q \in \mathit{Prop}.\ \lceil p * q \rceil \subseteq \lceil p \rceil * \lceil q \rceil$$

*Proof.* Assume $(l, s, \vartheta) \in \lceil p * q \rceil$. Then by assumption there exists $l', s'$ such that

$$(l', s', \vartheta) \in p * q \qquad\qquad (l', s')\ R\ (l, s)$$

By the definition of $*$ there thus exists $l_1', l_2'$ such that

$$l' = l_1' \bullet_{\mathit{LState}} l_2' \qquad (l_1', s', \vartheta) \in p \qquad (l_2', s', \vartheta) \in q$$

Hence $(l_1', s')\ R\ (l_1', s)$ and $(l_2', s')\ R\ (l_2', s)$ from which it follows that

$$(l_1', s, \vartheta) \in \lceil p \rceil \qquad\qquad (l_2', s, \vartheta) \in \lceil q \rceil$$

Lastly, since $l' \leq l$ it follows that there exists an $l''$ such that $l = l' \bullet l''$. By upwards-closure of $\lceil p \rceil$ it thus follows that $(l_1' \bullet l'', s, \vartheta) \in \lceil p \rceil$ and thus

$$(l, s, \vartheta) \in \lceil p \rceil * \lceil q \rceil.$$

□

**Lemma 32.**

$$\forall p \in \mathit{Prop}.\ \mathit{stable}(\lceil p \rceil)$$

*Proof.* Assume $(l_1, s_1, \vartheta) \in \lceil p \rceil$ and $(l_1, s_1)\ R\ (l_2, s_2)$. By definition of $\lceil - \rceil$ it follows that there exists $l_1', s_1'$ such that

$$(l_1', s_1', \vartheta) \in p \qquad\qquad (l_1', s_1')\ R\ (l_1, s_1)$$

Hence, by transitivity of $R$, $(l_1', s_1')\ R\ (l_2, s_2)$ and thus $(l_2, s_2, \vartheta) \in \lceil p \rceil$. □

**Lemma 33.**

$$\forall p \in \mathit{Prop}.\ p \subseteq \lceil p \rceil$$

*Proof.* Assume $(l, s, \vartheta) \in p$ then $(l, s, \vartheta) \in \lceil p \rceil$ as $(l, s)\ R\ (l, s)$ holds trivially. □

**Lemma 34.**

$$\forall p \in \mathit{Prop}.\ \mathit{stable}(p) \Rightarrow \lceil p \rceil \subseteq p$$

*Proof.* Assume $(l, s, \vartheta) \in \lceil p \rceil$. By definition of $\lceil - \rceil$ there exists $l', s'$ such that

$$(l', s', \vartheta) \in p \qquad\qquad (l', s')\ R\ (l, s)$$

Hence, by stability of $p$, $(l, s, \vartheta) \in p$. □

**Lemma 35.**

$$\forall p, q \in \mathit{Prop}.\ p \subseteq q \Rightarrow \lceil p \rceil \subseteq \lceil q \rceil$$

*Proof.* Assume $(l, s, \vartheta) \in \lceil p \rceil$. By definition of $\lceil - \rceil$ there exists $l', s'$ such that

$$(l', s', \vartheta) \in p \qquad\qquad (l', s')\ R\ (l, s)$$

Hence $(l', s', \vartheta) \in q$ and thus $(l, s, \vartheta) \in \lceil q \rceil$. □

*Separating Implication* $\boxed{-\!\!* \in \mathit{Prop} \times \mathit{Prop} \to \mathit{Prop} \in \mathcal{S}}$

$$p -\!\!* q \stackrel{\text{def}}{=} \{m \in \mathcal{M} \mid \forall m' \geq m.\ \forall m'' \in \mathcal{M}.\ (m' \bullet m'' \text{ defined} \wedge m'' \in p \Rightarrow (m' \bullet m'') \in q\}$$

**Lemma 36.**

$$\forall p, q \in \mathit{Prop}.\ \mathit{valid}(\triangleright(p -\!\!* q) \Rightarrow ((\triangleright p) -\!\!* (\triangleright q)))$$

*Proof.* Let $m \in \mathcal{M}$ such that $m \in \triangleright(p -\!\!* q)$. Hence

$$\triangleright(\forall m', m'' \in \mathcal{M}.\ m \leq m' \wedge m' \bullet m'' \text{ defined} \wedge m'' \in p \Rightarrow (m' \bullet m'') \in q)$$

and thus

$$\forall m', m'' \in \mathcal{M}.\ \triangleright\,(m \leq m') \wedge \triangleright(m' \bullet m'' \text{ defined}) \wedge m'' \in \triangleright p \Rightarrow (m' \bullet m'') \in \triangleright q$$

Hence, for all $m', m'' \in \mathcal{M}$ such that $m \leq m'$, $m' \bullet m''$ defined and $m'' \in \triangleright p$, it follows that $m' \bullet m'' \in \triangleright q$. □

**Lemma 37.**

$$\begin{aligned}&\forall r \in \Delta(\mathit{RId}).\ \forall I, J \in \Delta(\mathit{SId}) \to \mathit{Prop}.\ \forall x \in \Delta(\mathit{SId}).\\ &\quad \mathit{valid}(\mathit{rintr}(r, I) * \mathit{rintr}(r, J) \Rightarrow (\triangleright I(x) -\!\!* \triangleright J(x)))\end{aligned}$$

*Proof.* Let $(l, s, \vartheta) \in \mathcal{M}$ such that $(l, s, \vartheta) \in \mathit{rintr}(r, I) * \mathit{rintr}(r, J)$. Then by definition of $*$ and *rintr* it follows that

$$\forall \vartheta' \geq \vartheta.\ \mathit{app}(\vartheta(r))(x, \vartheta') = \triangleright(\lambda(l, s).\ I(x)(l, s, \vartheta')) = \triangleright(\lambda(l, s).\ J(x)(l, s, \vartheta'))$$

To show that $(l, s, \vartheta) \in \triangleright I(s) -\!\!* \triangleright J(x)$ let $(l', s', \vartheta'), (l'', s'', \vartheta'') \in \mathcal{M}$ such that

$$(l, s, \vartheta) \leq (l', s', \vartheta') \qquad (l', s', \vartheta') \bullet (l'', s'', \vartheta'') \text{ defined} \qquad (l'', s'', \vartheta'') \in \triangleright I(x)$$

Since $(l', s', \vartheta') \bullet (l'', s'', \vartheta'')$ is defined it follows that $\vartheta' = \vartheta''$ and thus that $\vartheta \leq \vartheta''$.

Hence, $(l'', s'', \vartheta'') \in \triangleright J(x)$ and thus, by upwards-closure,

$$(l', s', \vartheta') \bullet (l'', s'', \vartheta'') \in \triangleright J(x)$$

□

# 2 Program Logic

## 2.1 Syntax

$$
\begin{array}{rlll}
\text{Terms} \quad \mathsf{M,N,P,Q,S,T,F} & ::= & \lambda \mathsf{x} : \tau.\ \mathsf{M} \mid \mathsf{M}\ \mathsf{N} \mid \mathsf{x} & \\
& \mid & \bot \mid \top \mid \mathsf{M} \vee \mathsf{N} \mid \mathsf{M} \wedge \mathsf{M} \mid \mathsf{M} \Rightarrow \mathsf{N} & \\
& \mid & \forall \mathsf{x} : \tau.\ \mathsf{P} \mid \exists \mathsf{x} : \tau.\ \mathsf{P} \mid \mathsf{M} =_\tau \mathsf{N} & \\
& \mid & \mathsf{P} * \mathsf{Q} \mid \mathsf{P} \mathrel{-\!\!*} \mathsf{Q} \mid \mathsf{emp} & \\
& \mid & \mathsf{M.F} \mapsto \mathsf{N} \mid \mathsf{M}_\mathsf{F} \mapsto \mathsf{N} \mid \mathsf{M} \mapsto \mathsf{N.M} \mid \mathsf{M{:}N} & \\
& \mid & \mathsf{region}(\mathsf{R},\mathsf{M},\mathsf{N}) \mid \mathsf{rintr}(\mathsf{R},\mathsf{M}) \mid [\mathsf{M}]^\mathsf{R}_\mathsf{N} \mid \mathsf{stable}(\mathsf{P}) & \\
& \mid & \mathsf{P} \sqsubseteq^\mathsf{R} \mathsf{Q} \mid (\Delta).\{\mathsf{P}\}\langle \mathsf{s}\rangle\{\mathsf{Q}\} \mid (\Delta).\{\mathsf{P}\}\langle \mathsf{s}\rangle_\mathsf{R}\{\mathsf{Q}\} & \\
& \mid & \triangleright \mathsf{M} \mid \mathsf{fix}_\tau(\mathsf{M}) \mid \mathsf{guarded}_\tau(\mathsf{M}) & \\
& \mid & \mathsf{valid}(\mathsf{P}) \mid \mathsf{spec}(\mathsf{S}) & \\
& \mid & \mathsf{M.N} : (\Delta).\{\mathsf{P}\}\{\mathsf{x.Q}\} \mid \mathsf{M} : (\Delta).\{\mathsf{P}\}\{\mathsf{x.Q}\} & \\
& \mid & (\Delta).\{\mathsf{P}\}\bar{\mathsf{s}}\{\mathsf{Q}\} \mid (\Delta).\langle \mathsf{P}\rangle\bar{\mathsf{s}}\langle \mathsf{Q}\rangle^\mathsf{R} & \\
& \mid & \Delta_X(x) & x \in X, X \in obj(\mathrm{Sets})
\end{array}
$$

$$
\text{Types} \quad \tau,\sigma \quad ::= \quad 1 \mid \tau \to \sigma \mid \tau \times \sigma \mid \tau + \sigma \mid \mathcal{P}(\tau) \mid \Delta(X) \mid \mathsf{Prop} \mid \mathsf{Spec}
$$

*Well-formed types* $\boxed{\vdash \tau : \mathsf{Type}}$

$$
\frac{}{\vdash 1 : \mathsf{Type}} \qquad \frac{\vdash \tau : \mathsf{Type} \qquad \vdash \sigma : \mathsf{Type}}{\vdash \tau \times \sigma : \mathsf{Type}} \qquad \frac{\vdash \tau : \mathsf{Type} \qquad \vdash \sigma : \mathsf{Type}}{\vdash \tau + \sigma : \mathsf{Type}} \qquad \frac{\vdash \tau : \mathsf{Type}}{\vdash \mathcal{P}(\tau) : \mathsf{Type}}
$$

$$
\frac{\vdash \tau : \mathsf{Type} \qquad \vdash \sigma : \mathsf{Type}}{\vdash \tau \to \sigma : \mathsf{Type}} \qquad \frac{}{\vdash \mathsf{Prop} : \mathsf{Type}} \qquad \frac{}{\vdash \mathsf{Spec} : \mathsf{Type}} \qquad \frac{\begin{array}{c} x \in X \\ X \in obj(\mathrm{Sets}) \end{array}}{\vdash \Delta(X) : \mathsf{Type}}
$$

$$
\mathsf{Class} \stackrel{\text{def}}{=} \Delta(\mathit{CName}) \qquad \mathsf{Method} \stackrel{\text{def}}{=} \Delta(\mathit{MName}) \qquad \mathsf{Field} \stackrel{\text{def}}{=} \Delta(\mathit{FName}) \qquad \mathsf{Val} \stackrel{\text{def}}{=} \Delta(\mathit{Val})
$$

$$
\mathsf{Perm} \stackrel{\text{def}}{=} \Delta(\mathit{Perm}_n) \qquad \mathsf{Action} \stackrel{\text{def}}{=} \Delta(\mathit{AId}) \qquad \mathsf{Region} \stackrel{\text{def}}{=} \Delta(\mathit{RId})
$$

*Well-formed terms* $\boxed{\Gamma;\Delta \vdash \mathsf{M} : \tau}$

Lambda calculus typing

$$
\frac{\vdash \tau : \mathsf{Type} \qquad (\mathsf{x} : \tau) \in \Gamma}{\Gamma;\Delta \vdash \mathsf{x} : \tau} \qquad \frac{(\mathsf{x} : \mathsf{Val}) \in \Delta}{\Gamma;\Delta \vdash \mathsf{x} : \mathsf{Val}}
$$

$$
\frac{\Gamma,\mathsf{x} : \tau;\Delta \vdash \mathsf{M} : \sigma}{\Gamma;\Delta \vdash \lambda \mathsf{x} : \tau.\ \mathsf{M} : \tau \to \sigma} \qquad \frac{\Gamma;\Delta \vdash \mathsf{M} : \tau \to \sigma \qquad \Gamma;\Delta \vdash \mathsf{M} : \tau}{\Gamma;\Delta \vdash \mathsf{M}\ \mathsf{N} : \sigma}
$$

To distinguish variables in the logic and meta-logic, we mostly use sans-serif identifiers, such as $\mathsf{x}$ in the logic, and italic identifiers, such as $x$ in the meta-logic.

Assertion typing

$$\frac{}{\Gamma;\Delta \vdash \bot : \mathsf{Prop}} \qquad \frac{}{\Gamma;\Delta \vdash \top : \mathsf{Prop}}$$

$$\frac{\Gamma;\Delta \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma;\Delta \vdash \mathsf{Q} : \mathsf{Prop}}{\Gamma;\Delta \vdash \mathsf{P} \wedge \mathsf{Q} : \mathsf{Prop}} \qquad \frac{\Gamma;\Delta \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma;\Delta \vdash \mathsf{Q} : \mathsf{Prop}}{\Gamma;\Delta \vdash \mathsf{P} \vee \mathsf{Q} : \mathsf{Prop}}$$

$$\frac{\Gamma;\Delta \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma;\Delta \vdash \mathsf{Q} : \mathsf{Prop}}{\Gamma;\Delta \vdash \mathsf{P} \Rightarrow \mathsf{Q} : \mathsf{Prop}} \qquad \frac{\Gamma;\Delta \vdash \mathsf{M} : \tau \qquad \Gamma;\Delta \vdash \mathsf{N} : \tau}{\Gamma;\Delta \vdash \mathsf{M} =_\tau \mathsf{N} : \mathsf{Prop}}$$

$$\frac{\Gamma, \mathsf{x} : \tau;\Delta \vdash \mathsf{P} : \mathsf{Prop}}{\Gamma;\Delta \vdash \forall \mathsf{x} : \tau.\ \mathsf{P} : \mathsf{Prop}} \qquad \frac{\Gamma, \mathsf{x} : \tau;\Delta \vdash \mathsf{P} : \mathsf{Prop}}{\Gamma;\Delta \vdash \exists \mathsf{x} : \tau.\ \mathsf{P} : \mathsf{Prop}}$$

Constant sets

$$\frac{x \in X \qquad X \in obj(\mathrm{Sets})}{\Gamma;\Delta \vdash \Delta_X(x) : \Delta(X)}$$

Separation logic typing

$$\frac{}{\Gamma;\Delta \vdash \mathsf{emp} : \mathsf{Prop}} \qquad \frac{\Gamma;\Delta \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma;\Delta \vdash \mathsf{Q} : \mathsf{Prop} \qquad op \in \{*, -\!\!*\}}{\Gamma;\Delta \vdash \mathsf{P}\ op\ \mathsf{Q} : \mathsf{Prop}}$$

C$^\sharp$ typings

$$\frac{\mathsf{x} \in \Delta}{\Gamma;\Delta \vdash \mathsf{x} : \mathsf{Val}}$$

$$\frac{\Gamma;\Delta \vdash \mathsf{M} : \mathsf{Val} \qquad \Gamma;\Delta \vdash \mathsf{C} : \mathsf{Class}}{\Gamma;\Delta \vdash \mathsf{M} : \mathsf{C} : \mathsf{Prop}}$$

$$\frac{\Gamma;\Delta \vdash \mathsf{M} : \mathsf{Val} \qquad \Gamma;\Delta \vdash \mathsf{F} : \mathsf{Field} \qquad \Gamma;\Delta \vdash \mathsf{N} : \mathsf{Val}}{\Gamma;\Delta \vdash \mathsf{M}.\mathsf{F} \mapsto \mathsf{N} : \mathsf{Prop}}$$

$$\frac{\Gamma;\Delta \vdash \mathsf{M} : \mathsf{Val} \qquad \Gamma;\Delta \vdash \mathsf{F} : \mathsf{Field} \qquad \Gamma;\Delta \vdash \mathsf{N} : \mathsf{Val}}{\Gamma;\Delta \vdash \mathsf{M}_\mathsf{F} \mapsto \mathsf{N} : \mathsf{Prop}}$$

$$\frac{\Gamma;\Delta \vdash \mathsf{M}_1 : \mathsf{Val} \qquad \Gamma;\Delta \vdash \mathsf{N} : \mathsf{Val} \qquad \Gamma;\Delta \vdash \mathsf{M}_2 : \mathsf{Method}}{\Gamma;\Delta \vdash \mathsf{M}_1 \mapsto \mathsf{N}.\mathsf{M}_2 : \mathsf{Prop}}$$

Specification & assertion embedding

$$\frac{\Gamma \vdash \mathsf{S} : \mathsf{Spec}}{\Gamma; \Delta \vdash \mathsf{spec}(\mathsf{S}) : \mathsf{Prop}} \qquad \frac{\Gamma; - \vdash \mathsf{P} : \mathsf{Prop}}{\Gamma \vdash \mathsf{valid}(\mathsf{P}) : \mathsf{Spec}}$$

$$\mathsf{x} \mapsto (\Delta).\{\mathsf{P}\}\{\mathsf{r}.\mathsf{Q}\} \stackrel{\text{def}}{=} \exists \mathsf{y} : \mathsf{Val}.\ \exists \mathsf{m} : \mathsf{Method}.\ \exists \mathsf{C} : \mathsf{Class}.\ \mathsf{x} \mapsto \mathsf{y}.\mathsf{m} * \mathsf{y} : \mathsf{C} * \mathsf{spec}(\mathsf{C}.\mathsf{m} : (\Delta).\{\mathsf{P}\}\{\mathsf{r}.\mathsf{Q}\})$$

Guarded recursion typing

$$\frac{\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop}}{\Gamma; \Delta \vdash \triangleright \mathsf{P} : \mathsf{Prop}}$$

$$\frac{\Gamma; \Delta \vdash \mathsf{M} : (\tau \to \mathsf{Prop}) \to (\tau \to \mathsf{Prop})}{\Gamma; \Delta \vdash \mathsf{guarded}_\tau(\mathsf{M}) : \mathsf{Spec}}$$

Concurrent abstract predicates typing

$$\frac{\Gamma; \Delta \vdash \mathsf{R} : \mathsf{Region} \qquad \Gamma; \Delta \vdash \mathsf{M} : \mathcal{P}(\mathsf{SId}) \qquad \Gamma; \Delta \vdash \mathsf{N} : \mathsf{Action} \to \mathcal{P}(\mathsf{SId} \times \mathsf{SId})}{\Gamma; \Delta \vdash \mathsf{region}(\mathsf{M}, \mathsf{N}, \mathsf{R}) : \mathsf{Prop}}$$

$$\frac{\Gamma; \Delta \vdash \mathsf{A} : \mathsf{Action} \qquad \Gamma; \Delta \vdash \mathsf{R} : \mathsf{Region} \qquad \Gamma; \Delta \vdash \mathsf{P} : \mathsf{Perm}}{\Gamma; \Delta \vdash [\mathsf{A}]^{\mathsf{R}}_{\mathsf{P}} : \mathsf{Prop}}$$

$$\frac{\Gamma; \Delta \vdash \mathsf{R} : \mathsf{Region} \qquad \Gamma; \Delta \vdash \mathsf{I} : \mathsf{SId} \to \mathsf{Prop}}{\Gamma; \Delta \vdash \mathsf{rintr}(\mathsf{I}, \mathsf{R}) : \mathsf{Prop}}$$

We use $\mathsf{region}(\mathsf{M}, \mathsf{N}, \mathsf{I}, \mathsf{R})$ as shorthand for $\mathsf{region}(\mathsf{M}, \mathsf{N}, \mathsf{R}) * \mathsf{rintr}(\mathsf{I}, \mathsf{R})$.

$$\frac{\Gamma \vdash \mathsf{P} : \mathsf{Prop}}{\Gamma \vdash \mathsf{stable}(\mathsf{P}) : \mathsf{Spec}} \qquad \frac{\Gamma \vdash \mathsf{R} : \mathcal{P}(\mathsf{Region}) \qquad \Gamma \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma \vdash \mathsf{Q} : \mathsf{Prop}}{\Gamma \vdash \mathsf{P} \sqsubseteq^{\mathsf{R}} \mathsf{Q} : \mathsf{Spec}}$$

Specification typings

$$\frac{}{\Gamma \vdash \bot : \mathsf{Spec}} \qquad \frac{}{\Gamma \vdash \top : \mathsf{Spec}} \qquad \frac{\Gamma \vdash \mathsf{M} : \tau \qquad \Gamma \vdash \mathsf{N} : \tau}{\Gamma \vdash \mathsf{M} =_\tau \mathsf{N} : \mathsf{Spec}}$$

$$\frac{\Gamma \vdash \mathsf{S} : \mathsf{Spec} \qquad \Gamma \vdash \mathsf{T} : \mathsf{Spec}}{\Gamma \vdash \mathsf{S} \wedge \mathsf{T} : \mathsf{Spec}} \qquad \frac{\Gamma \vdash \mathsf{S} : \mathsf{Spec} \qquad \Gamma \vdash \mathsf{T} : \mathsf{Spec}}{\Gamma \vdash \mathsf{S} \vee \mathsf{T} : \mathsf{Spec}}$$

$$\frac{\Gamma \vdash \mathsf{S} : \mathsf{Spec} \qquad \Gamma \vdash \mathsf{T} : \mathsf{Spec}}{\Gamma \vdash \mathsf{S} \Rightarrow \mathsf{T} : \mathsf{Spec}} \qquad \frac{\Gamma \vdash \mathsf{S} : \mathsf{Spec}}{\Gamma \vdash \triangleright \mathsf{T} : \mathsf{Spec}}$$

$$\frac{\Gamma, \mathsf{x} : \tau \vdash \mathsf{S} : \mathsf{Spec}}{\Gamma \vdash \forall \mathsf{x} : \tau.\ \mathsf{S} : \mathsf{Spec}} \qquad \frac{\Gamma, \mathsf{x} : \tau \vdash \mathsf{S} : \mathsf{Spec}}{\Gamma \vdash \exists \mathsf{x} : \tau.\ \mathsf{S} : \mathsf{Spec}}$$

Hoare typings

$$\frac{\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma; \Delta \vdash \mathsf{Q} : \mathsf{Prop} \qquad \mathsf{FV}(\mathsf{s}) \subseteq \mathsf{vars}(\Delta)}{\Gamma \vdash (\Delta).\{\mathsf{P}\}\mathsf{s}\{\mathsf{Q}\} : \mathsf{Spec}}\ \textsc{HoareS}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathsf{C} : \mathsf{Class} \qquad \Gamma \vdash \mathsf{M} : \mathsf{Method} \\ \Gamma; \Delta, \mathtt{this} \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma; \Delta, \mathtt{this}, \mathsf{x} \vdash \mathsf{Q} : \mathsf{Prop}\end{array}}{\Gamma \vdash \mathsf{C.M} : (\Delta).\{\mathsf{P}\}\{\mathsf{x}.\mathsf{Q}\} : \mathsf{Spec}}\ \textsc{HoareM}$$

$$\frac{\Gamma \vdash \mathsf{C} : \mathsf{Class} \qquad \Gamma \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma; \mathsf{x} \vdash \mathsf{Q} : \mathsf{Prop}}{\Gamma \vdash \mathsf{C} : \{\mathsf{P}\}\{\mathsf{x}.\mathsf{Q}\} : \mathsf{Spec}}\ \textsc{HoareC}$$

$$\frac{\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma; \Delta \vdash \mathsf{Q} : \mathsf{Prop} \qquad \Gamma \vdash \mathsf{R} : \mathcal{P}(\mathsf{Region})}{\Gamma \vdash (\Delta).\langle \mathsf{P} \rangle \mathsf{s} \langle \mathsf{Q} \rangle^{\mathsf{R}} : \mathsf{Spec}}\ \textsc{HoareA}$$

## 2.2 Logics

The specification logic is given by the specification entailment judgment

$$\Gamma \mid \Phi \vdash \mathsf{S},$$

where $\mathsf{S}$ is a specification and $\Phi$ is a specification context:

$$\Phi ::= \Phi, \mathsf{S} \mid \varepsilon$$

such that $\Gamma \vdash \mathsf{S} : \mathsf{Spec}$ and $\Gamma \vdash \mathsf{T} : \mathsf{Spec}$ for each assumption $\mathsf{T}$ in $\Phi$.

The assertion logic is given by the assertion entailment judgment

$$\Gamma; \Delta \mid \Phi \mid \mathsf{P} \vdash \mathsf{Q}$$

where $\mathsf{P}$ and $\mathsf{Q}$ are assertions, such that $\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop}$ and $\Gamma; \Delta \vdash \mathsf{Q} : \mathsf{Prop}$ and $\Gamma \vdash \mathsf{T} : \mathsf{Spec}$ for each assumption $\mathsf{T}$ in $\Phi$. The assertion entailment includes the specification context $\Phi$, to allow the use of assertion assumptions embedded in specifications.

### 2.2.1 Assertion logic

The assertion logic includes a standard intuitionistic higher-order separation logic.

$$\frac{}{\Gamma; \Delta \mid \Phi \mid \bot \vdash \mathsf{P}} \qquad \frac{}{\Gamma; \Delta \mid \Phi \mid \mathsf{P} \vdash \top}$$

$$\frac{}{\Gamma; \Delta \mid \Phi \mid \mathsf{P} * \mathsf{Q} \vdash \mathsf{P}} \qquad \frac{\Gamma; \Delta \mid \Phi \mid \mathsf{P} \vdash \mathsf{Q} \qquad \Gamma; \Delta \mid \Phi \mid \mathsf{Q} \vdash \mathsf{R}}{\Gamma; \Delta \mid \Phi \mid \mathsf{P} \vdash \mathsf{R}}$$

$$\frac{\Gamma; \Delta \mid \Phi \mid \mathsf{P} \vdash \mathsf{R} \qquad \Gamma; \Delta \mid \Phi \mid \mathsf{Q} \vdash \mathsf{R}}{\Gamma; \Delta \mid \Phi \mid \mathsf{P} \vee \mathsf{Q} \vdash \mathsf{R}}$$

$$\frac{\Gamma; \Delta \mid \Phi \mid \mathsf{P} \vee \mathsf{Q} \vdash \mathsf{R}}{\Gamma; \Delta \mid \Phi \mid \mathsf{P} \vdash \mathsf{R}} \qquad \frac{\Gamma; \Delta \mid \Phi \mid \mathsf{P} \vee \mathsf{Q} \vdash \mathsf{R}}{\Gamma; \Delta \mid \Phi \mid \mathsf{Q} \vdash \mathsf{R}}$$

$$\frac{\Gamma; \Delta \mid \Phi \mid \mathsf{P} \wedge \mathsf{Q} \vdash \mathsf{R}}{\Gamma; \Delta \mid \Phi \mid \mathsf{P} \vdash \mathsf{Q} \Rightarrow \mathsf{R}} \qquad \frac{\Gamma; \Delta \mid \Phi \mid \mathsf{P} \vdash \mathsf{Q} \Rightarrow \mathsf{R}}{\Gamma; \Delta \mid \Phi \mid \mathsf{P} \wedge \mathsf{Q} \vdash \mathsf{R}}$$

$$\frac{\Gamma; \Delta \mid \Phi \mid \mathsf{P} \vdash \mathsf{Q} \mathrel{-\!\!*} \mathsf{R}}{\Gamma; \Delta \mid \Phi \mid \mathsf{P} * \mathsf{Q} \vdash \mathsf{R}} \qquad \frac{\Gamma; \Delta \mid \Phi \mid \mathsf{P} * \mathsf{Q} \vdash \mathsf{R}}{\Gamma; \Delta \mid \Phi \mid \mathsf{P} \vdash \mathsf{Q} \mathrel{-\!\!*} \mathsf{R}}$$

$$\frac{\Gamma; - \mid \Phi \mid \top \vdash \mathsf{P}}{\Gamma \mid \Phi \vdash \mathsf{valid}(\mathsf{P})} \qquad \frac{\Gamma \mid \Phi \vdash \mathsf{valid}(\mathsf{P})}{\Gamma; - \mid \Phi \mid \top \vdash \mathsf{P}}$$

### 2.2.2 Specification logic

The specification logic includes a standard intuitionistic higher-order logic.

$$\frac{}{\Gamma \mid \Phi, \bot \vdash \mathsf{S}} \qquad \frac{}{\Gamma \mid \Phi \vdash \top} \qquad \frac{\Gamma \mid \Phi, \triangleright \mathsf{S} \vdash \mathsf{S}}{\Gamma \mid \Phi \vdash \mathsf{S}}$$

$$\frac{\Gamma \mid \Phi \vdash \mathsf{S} \qquad \Gamma \mid \Phi \vdash \mathsf{T}}{\Gamma \mid \Phi \vdash \mathsf{S} \wedge \mathsf{T}} \qquad \frac{\Gamma \mid \Phi \vdash \mathsf{S}}{\Gamma \mid \Phi \vdash \mathsf{S} \vee \mathsf{T}} \qquad \frac{\Gamma \mid \Phi \vdash \mathsf{T}}{\Gamma \mid \Phi \vdash \mathsf{S} \vee \mathsf{T}}$$

$$\frac{\Gamma \mid \Phi \vdash \mathsf{S} \wedge \mathsf{T}}{\Gamma \mid \Phi \vdash \mathsf{S}} \qquad \frac{\Gamma \mid \Phi \vdash \mathsf{S} \wedge \mathsf{T}}{\Gamma \mid \Phi \vdash \mathsf{T}} \qquad \frac{\Gamma \mid \Phi \vdash \mathsf{S}_1 \vee \mathsf{S}_2 \qquad \Gamma \mid \Phi, \mathsf{S}_i \vdash \mathsf{T}}{\Gamma \mid \Phi \vdash \mathsf{T}}$$

$$\frac{\Gamma \mid \Phi, \mathsf{S} \vdash \mathsf{T}}{\Gamma \mid \Phi \vdash \mathsf{S} \Rightarrow \mathsf{T}} \qquad \frac{\Gamma \mid \Phi \vdash \mathsf{S} \Rightarrow \mathsf{T} \qquad \Gamma \mid \Phi \vdash \mathsf{S}}{\Gamma \mid \Phi \vdash \mathsf{T}}$$

$$\frac{\Gamma \mid \Phi \vdash \forall \mathsf{x} : \tau.\ \mathsf{S} \qquad \Gamma \vdash \mathsf{M} : \tau}{\Gamma \mid \Phi \vdash \mathsf{S}[\mathsf{M}/\mathsf{x}]} \qquad \frac{\Gamma, \mathsf{x} : \tau \mid \Phi \vdash \mathsf{S} \qquad \mathsf{x} \notin \mathsf{FV}(\Phi)}{\Gamma \mid \Phi \vdash \forall \mathsf{x} : \tau.\ \mathsf{S}}$$

$$\frac{\Gamma \mid \Phi \vdash \mathsf{S}[\mathsf{M}/\mathsf{x}] \qquad \Gamma \vdash \mathsf{M} : \tau}{\Gamma \mid \Phi \vdash \exists \mathsf{x} : \tau.\ \mathsf{S}} \qquad \frac{\Gamma, \mathsf{x} : \tau \mid \Phi, \mathsf{S} \vdash \mathsf{T} \qquad \mathsf{x} \notin \mathsf{FV}(\Phi, \mathsf{T})}{\Gamma \mid \Phi, \exists \mathsf{x} : \tau.\ \mathsf{S} \vdash \mathsf{T}}$$

$$\frac{\Gamma \mid \mathsf{M} : \tau}{\Gamma \mid \Phi \vdash \mathsf{M} =_\tau \mathsf{M}} \qquad \frac{\Gamma \mid \Phi \vdash \mathsf{M} =_\tau \mathsf{N} \qquad \Gamma \mid \Phi \vdash \mathsf{S}[\mathsf{M}/x]}{\Gamma \mid \Phi \vdash \mathsf{S}[\mathsf{N}/x]}$$

Later operator

$$\frac{\Gamma \mid \Phi, \triangleright \mathsf{S} \vdash \mathsf{S}}{\Gamma \mid \Phi \vdash \mathsf{S}}\ \textsc{SLob} \qquad \frac{}{\Gamma \mid \Phi \vdash \mathsf{S} \Rightarrow \triangleright \mathsf{S}}\ \textsc{SMono}$$

$$\frac{\Gamma \mid \Phi \vdash \langle x.f \mapsto v \rangle\ c\ \langle \mathsf{Q} \rangle^A}{\Gamma \mid \Phi \vdash \langle \triangleright (x.f \mapsto v) \rangle\ c\ \langle \mathsf{Q} \rangle^A}\ \textsc{LPoints}$$

$$\frac{op \in \{\wedge, \vee, *\}}{\Gamma; \Delta \mid \Phi \mid \triangleright(\mathsf{P}\ op\ \mathsf{Q}) \dashv\vdash (\triangleright \mathsf{P})\ op\ (\triangleright \mathsf{Q})}\ \textsc{LBin}$$

$$\frac{}{\Gamma; \Delta \mid \Phi \mid \triangleright(\mathsf{P} \Rightarrow \mathsf{Q}) \vdash (\triangleright\mathsf{P}) \Rightarrow (\triangleright\mathsf{Q})}\ \textsc{LImpl}$$

$$\frac{}{\Gamma; \Delta \mid \Phi \mid \triangleright(\mathsf{P} \mathrel{-\!\!*} \mathsf{Q}) \vdash (\triangleright\mathsf{P}) \mathrel{-\!\!*} (\triangleright\mathsf{Q})}\ \textsc{LWand}$$

$$\frac{Q \in \{\forall, \exists\}}{\Gamma; \Delta \mid \Phi \mid \triangleright(Qx : \tau.\ \mathsf{P}(x)) \dashv\vdash Qx : \tau.\ \triangleright\, \mathsf{P}(x)}\ \textsc{LQuant}$$

Specification embedding

$$\frac{\Gamma \mid \Phi, \mathsf{S} \vdash \mathsf{P} \sqsubseteq^{A} \mathsf{Q}}{\Gamma \mid \Phi \vdash \mathsf{P} * \mathsf{spec}(\mathsf{S}) \sqsubseteq^{A} \mathsf{Q}}\ \textsc{VS} \qquad \frac{\Gamma \mid \Phi, \mathsf{S} \vdash \{\mathsf{P}\}\ c\ \{\mathsf{Q}\}}{\Gamma \mid \Phi \vdash \{\mathsf{P} * \mathsf{spec}(\mathsf{S})\}\ c\ \{\mathsf{Q}\}}\ \textsc{HS}$$

$$\frac{\Gamma \mid \Phi, \mathsf{S} \vdash \langle \mathsf{P} \rangle\ c\ \langle \mathsf{Q} \rangle^{A}}{\Gamma \mid \Phi \vdash \langle \mathsf{P} * \mathsf{spec}(\mathsf{S}) \rangle\ c\ \langle \mathsf{Q} \rangle^{A}}\ \textsc{AS}$$

View-shifts

$$\frac{\Gamma \mid \Phi \vdash \mathsf{P} \sqsubseteq^{A} \mathsf{Q} \qquad \Gamma \mid \Phi \vdash \mathsf{Q} \sqsubseteq^{A} \mathsf{R}}{\Gamma \mid \Phi \vdash \mathsf{Q} \sqsubseteq^{A} \mathsf{R}}\ \textsc{VTrans} \qquad \frac{\Gamma \mid \Phi \mid \mathsf{P} \vdash \mathsf{Q}}{\Gamma \mid \Phi \vdash \mathsf{P} \sqsubseteq^{A} \mathsf{Q}}\ \textsc{VImpl}$$

$$\frac{\begin{array}{c}\Gamma \mid \Phi \vdash x \in X \qquad \Gamma \mid \Phi \vdash \forall n \in C.\ \forall s.\ up(I(n)(s)) \\ \Gamma \mid \Phi \vdash A \text{ and } B \text{ are finite} \qquad \Gamma \mid \Phi \vdash C \text{ is infinite} \\ \Gamma \mid \Phi \vdash \forall n \in C.\ \mathsf{P} * \circledast_{\alpha \in A} [\alpha]_1^n \Rightarrow \triangleright I(n)(x) \\ \Gamma \mid \Phi \vdash \forall n \in C.\ \forall s.\ \mathsf{stable}(I(n)(s)) \qquad \Gamma \mid \Phi \vdash A \cap B = \emptyset\end{array}}{\Gamma \mid \Phi \vdash \mathsf{P} \sqsubseteq^{C} \exists n \in C.\ \mathsf{region}(X, T, I(n), n) * \circledast_{\alpha \in B} [\alpha]_1^n}\ \textsc{VAlloc}$$

$$\frac{\begin{array}{c}\Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{P}) \qquad \Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{Q}) \\ \Gamma \mid \Phi \vdash n \in A \qquad \Gamma \mid \Phi \vdash \forall x \in X.\ f(x) \in Y \\ \Gamma \mid \Phi \vdash \forall x \in X.\ (x, f(x)) \in T(\alpha) \vee f(x) = x \\ \Gamma \mid \Phi \vdash \forall x \in X.\ \mathsf{P} * \triangleright I(x) * [\alpha]_\pi^n \sqsubseteq^{A \setminus \{n\}} \mathsf{Q} * \triangleright I(f(x))\end{array}}{\Gamma \mid \Phi \vdash \mathsf{region}(X, T, I, n) * \mathsf{P} * [\alpha]_\pi^n \sqsubseteq^{A} \mathsf{region}(Y, T, I, n) * \mathsf{Q}}\ \textsc{VOpen}$$

### 2.2.3 Atomic commands

$$\frac{\Gamma, \Delta \vdash \mathsf{P}, \mathsf{Q} : \mathsf{Prop} \qquad \mathsf{atomic(s)} \qquad \Gamma \mid \Phi \vdash (\Delta).\langle \mathsf{P} \rangle \mathsf{s} \langle \mathsf{Q} \rangle^{\mathsf{RId}}}{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}\}\mathsf{s}\{\mathsf{Q}\}} \qquad \frac{\Gamma, \Delta \vdash \mathsf{P}, \mathsf{Q} : \mathsf{Prop} \qquad \Gamma, \Delta \vdash A, B : \mathcal{P}(\mathsf{RId}) \qquad \Gamma \mid \Phi \vdash (\Delta).\langle \mathsf{P} \rangle \mathsf{s} \langle \mathsf{Q} \rangle^{A \backslash B}}{\Gamma \mid \Phi \vdash (\Delta).\langle \mathsf{P} \rangle \mathsf{s} \langle \mathsf{Q} \rangle^{A}}$$

$$\frac{\begin{array}{c}\Gamma, \Delta \mid \Phi \vdash \mathsf{stable}(\mathsf{P}) \qquad \Gamma, \Delta \mid \Phi \vdash \forall y.\ \mathsf{stable}(\mathsf{Q}(y)) \\ \Gamma, \Delta \mid \Phi \vdash n \in C \qquad \Gamma, \Delta \mid \Phi \vdash \forall x \in X.\ (x, f(x)) \in \overline{T(A)} \vee f(x) = x \\ \Gamma \mid \Phi \vdash \forall x \in X.\ (\Delta).\langle \mathsf{P} * \circledast_{\alpha \in A} [\alpha]^{\mathsf{n}}_{g(\alpha)} * \triangleright I(x) \rangle\ c\ \langle \mathsf{Q}(x) * \triangleright I(f(x)) \rangle^{C \backslash \{n\}}\end{array}}{\begin{array}{c}\Gamma \mid \Phi \vdash (\Delta).\ \langle \mathsf{P} * \circledast_{\alpha \in A} [\alpha]^{\mathsf{n}}_{g(\alpha)} * \mathsf{region}(X, T, I, n) \rangle \\ c \\ \langle \exists x.\ \mathsf{Q}(x) * \mathsf{region}(\{f(x)\}, T, I, n) \rangle^{C}\end{array}}\ \textsc{Atomic}$$

$$\frac{}{\begin{array}{l}\Gamma \mid \Phi \vdash (\Delta).\langle \mathsf{y.f} \mapsto \mathsf{M} \rangle \\ \qquad \mathsf{x} = \mathtt{CAS}(\mathsf{y.f}, \mathsf{o}, \mathsf{n}) \\ \qquad \langle (\mathsf{x} = \mathsf{y} * \mathsf{o} = \mathsf{M} * \mathsf{y.f} \mapsto \mathsf{n}) \vee (\mathsf{x} = \mathsf{null} * \mathsf{o} \neq \mathsf{M} * \mathsf{y.f} \mapsto \mathsf{M}) \rangle^{\emptyset}\end{array}}\ \textsc{ACas}$$

$$\frac{\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma, \Delta \mid \Phi \vdash \mathsf{stable}(\mathsf{P}) \qquad \mathsf{x}, \mathsf{y} \in \mathsf{vars}(\Delta)}{\Gamma \mid \Phi \vdash (\Delta).\langle \mathsf{P}[\mathsf{y}/\mathsf{x}] \rangle \mathsf{x} = \mathsf{y} \langle \mathsf{P} \rangle}\ \textsc{AAsign}$$

$$\frac{\mathsf{x}, \mathsf{y} \in \mathsf{vars}(\Delta)}{\Gamma \mid \Phi \vdash (\Delta).\langle \mathsf{x.f} \mapsto \_ \rangle \mathsf{x.f} = \mathsf{y} \langle \mathsf{x.f} \mapsto \mathsf{y} \rangle}\ \textsc{AWrite}$$

$$\frac{\Gamma; - \vdash \mathsf{M} : \mathsf{Val} \qquad \mathsf{x}, \mathsf{y} \in \mathsf{vars}(\Delta)}{\Gamma \mid \Phi \vdash (\Delta).\langle \mathsf{x.f} \mapsto \mathsf{M} \rangle \mathsf{y} = \mathsf{x.f} \langle \mathsf{x.f} \mapsto \mathsf{M} \wedge \mathsf{y} =_{\mathsf{Val}} \mathsf{M} \rangle}\ \textsc{ARead}$$

$$\frac{}{\mathsf{atomic}(\mathsf{x} = \mathtt{CAS}(\mathsf{y.f}, \mathsf{o}, \mathsf{n}))} \qquad \frac{}{\mathsf{atomic}(\mathsf{x} = \mathsf{y})} \qquad \frac{}{\mathsf{atomic}(\mathsf{x.f} = \mathsf{y})} \qquad \frac{}{\mathsf{atomic}(\mathsf{x} = \mathsf{y.f})}$$

Stability

$$\frac{\Gamma \mid \Phi \vdash \forall \alpha \notin A.\ \forall x \in X.\ T(\alpha)(x) \subseteq X}{\Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{region}(X, T, n) * \circledast_{\alpha \in A} [\alpha]^{n}_{1})}$$

$$\frac{}{\Gamma \mid \Phi \vdash \mathsf{stable}(\top)} \qquad \frac{}{\Gamma \mid \Phi \vdash \mathsf{stable}(\bot)} \qquad \frac{}{\Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{emp})}$$

$$\frac{}{\Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{M.F} \mapsto \mathsf{N})} \qquad \frac{}{\Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{M}_\mathsf{F} \mapsto \mathsf{N})}$$

$$\frac{}{\Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{M}_1 \mapsto \mathsf{N}.\mathsf{M}_2)} \qquad \frac{}{\Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{M} : \mathsf{N})}$$

$$\frac{\Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{P}) \qquad \Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{Q})}{\Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{P} \vee \mathsf{Q})} \qquad \frac{\Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{P}) \qquad \Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{Q})}{\Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{P} \wedge \mathsf{Q})}$$

$$\frac{\Gamma \vdash \mathsf{M}, \mathsf{N} : \tau}{\Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{M} =_\tau \mathsf{N})} \qquad \frac{\Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{P}) \qquad \Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{Q})}{\Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{P} * \mathsf{Q})}$$

$$\frac{\Gamma \mid \Phi \vdash \forall \mathsf{x} : \tau.\ \mathsf{stable}(\mathsf{P}(\mathsf{x}))}{\Gamma \mid \Phi \vdash \mathsf{stable}(\forall \mathsf{x} : \tau.\ \mathsf{P}(\mathsf{x}))} \qquad \frac{\Gamma \mid \Phi \vdash \forall \mathsf{x} : \tau.\ \mathsf{stable}(\mathsf{P}(\mathsf{x}))}{\Gamma \mid \Phi \vdash \mathsf{stable}(\exists \mathsf{x} : \tau.\ \mathsf{P}(\mathsf{x}))}$$

$$\frac{\Gamma \vdash \mathsf{I} : \mathsf{Action} \to \mathcal{P}(\mathsf{SId} \times \mathsf{SId}) \qquad \Gamma \vdash \mathsf{N} : \mathsf{RId}}{\Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{rintr}(\mathsf{I}, \mathsf{N}))} \qquad \frac{\Gamma \vdash \mathsf{S} : \mathsf{Spec}}{\Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{spec}(\mathsf{S}))}$$

Frame rules

$$\frac{\Gamma \mid \Phi \vdash \mathsf{P} \sqsubseteq^A \mathsf{Q} \qquad \Gamma \mid \Phi \mid \mathsf{stable}(\mathsf{R})}{\Gamma \mid \Phi \vdash \mathsf{P} * \mathsf{R} \sqsubseteq^A \mathsf{Q} * \mathsf{R}}\ \textsc{VFrame}$$

$$\frac{\Gamma \mid \Phi \vdash (\Delta).\langle \mathsf{P} \rangle\ c\ \langle \mathsf{Q} \rangle^A \qquad \Gamma, \Delta \mid \Phi \vdash \mathsf{stable}(\mathsf{R})}{\Gamma \mid \Phi \vdash (\Delta).\langle \mathsf{P} * \triangleright \mathsf{R} \rangle\ c\ \langle \mathsf{Q} * \mathsf{R} \rangle^A}\ \textsc{AFrame}$$

$$\frac{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}\}\ c\ \{\mathsf{Q}\} \qquad \Gamma, \Delta \mid \Phi \vdash \mathsf{stable}(\mathsf{R})}{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P} * \mathsf{R}\}\ c\ \{\mathsf{Q} * \mathsf{R}\}}\ \textsc{Frame}$$

Rules of consequence

$$\frac{\Gamma, \Delta \mid \Phi \vdash \mathsf{P}_1 \sqsubseteq^A \mathsf{P}_2 \qquad \Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}_2\}\ c\ \{\mathsf{Q}_2\} \qquad \Gamma, \Delta \mid \Phi \vdash \mathsf{Q}_2 \sqsubseteq^B \mathsf{Q}_1}{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}_1\}\ c\ \{\mathsf{Q}_1\}}$$

$$\frac{\Gamma, \Delta \mid \Phi \vdash \mathsf{P}_1 \sqsubseteq^A \mathsf{P}_2 \qquad \Gamma \mid \Phi \vdash (\Delta).\langle\mathsf{P}_2\rangle\ c\ \langle\mathsf{Q}_2\rangle^A \qquad \Gamma, \Delta \mid \Phi \vdash \triangleright(\mathsf{Q}_2 \sqsubseteq^A \mathsf{Q}_1)}{\Gamma \mid \Phi \vdash (\Delta).\langle\mathsf{P}_1\rangle\ c\ \langle\mathsf{Q}_1\rangle^A}$$

Sequential composition

$$\frac{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}\}\mathsf{s}_1\{\mathsf{Q}\} \qquad \Gamma \mid \Phi \vdash (\Delta).\{\mathsf{Q}\}\mathsf{s}_2\{\mathsf{R}\}}{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}\}\mathsf{s}_1; \mathsf{s}_2\{\mathsf{R}\}}$$

Primitive statements

$$\frac{\begin{array}{c}\Gamma, \bar{\mathsf{z}}, \mathtt{this}, \mathsf{ret} \mid \Phi \vdash \mathsf{stable}(\mathsf{P}) \wedge \mathsf{stable}(\mathsf{Q}) \\ \Gamma \mid \Phi \vdash \triangleright(\mathsf{C}.\mathsf{m} : (\bar{\mathsf{z}}).\{\mathsf{P}\}\{\mathsf{ret}.\mathsf{Q}\}) \qquad \Gamma \vdash \mathsf{C} : \mathsf{Class}\end{array}}{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}[\bar{\mathsf{u}}/\bar{\mathsf{z}}, \mathsf{y}/\mathtt{this}] * \mathsf{y} : \mathsf{C}\}\mathsf{x} = \mathsf{y}.\mathsf{m}(\bar{\mathsf{u}})\{\mathsf{Q}[\bar{\mathsf{u}}/\bar{\mathsf{z}}, \mathsf{y}/\mathtt{this}, \mathsf{x}/\mathsf{ret}]\}}$$

$$\frac{\begin{array}{c}\Gamma, \bar{\mathsf{z}}, \mathtt{this}, \mathsf{ret} \mid \Phi \vdash \mathsf{stable}(\mathsf{P}) \wedge \mathsf{stable}(\mathsf{Q}) \\ \Gamma \mid \Phi \vdash \triangleright(\mathsf{C}.\mathsf{m} : (\bar{\mathsf{z}}).\{\mathsf{P}\}\{\mathsf{ret}.\mathsf{Q}\}) \qquad \Gamma \vdash \mathsf{C} : \mathsf{Class}\end{array}}{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}[\bar{\mathsf{u}}/\bar{\mathsf{z}}, \mathsf{y}/\mathtt{this}] * \mathsf{y} \mapsto \mathsf{z}.\mathsf{m} * \mathsf{z} : \mathsf{C}\}\mathsf{x} = \mathsf{y}(\bar{\mathsf{u}})\{\mathsf{Q}[\bar{\mathsf{u}}/\bar{\mathsf{z}}, \mathsf{y}/\mathtt{this}, \mathsf{x}/\mathsf{ret}]\}}$$

$$\frac{\begin{array}{c}\Gamma, \mathtt{this}, \mathsf{ret} \mid \Phi \vdash \mathsf{stable}(\mathsf{P}) \wedge \mathsf{stable}(\mathsf{Q}) \\ \Gamma \mid \Phi \vdash \triangleright(\mathsf{C} : \{\mathsf{P}\}\{\mathsf{ret}.\mathsf{Q}\})\end{array}}{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}\}\mathsf{x} = \mathtt{new}\ \mathsf{C}()\{\mathsf{Q}[\mathsf{x}/\mathsf{ret}]\}}$$

$$\frac{\Gamma \vdash \mathsf{C} : \mathsf{Class} \qquad \mathsf{x}, \mathsf{y} \in \mathsf{vars}(\Delta)}{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{emp}\}\mathsf{x} = \mathsf{y}.\mathsf{m}\{\mathsf{x} \mapsto \mathsf{y}.\mathsf{m}\}}$$

$$\frac{\begin{array}{c}\Gamma, \mathtt{this}, \mathsf{ret} \mid \Phi \vdash \mathsf{stable}(\mathsf{P}) \wedge \mathsf{stable}(\mathsf{Q}) \\ \Gamma \mid \Phi \vdash \triangleright(\mathsf{C}.\mathsf{m} : (-).\{\mathsf{P}\}\{\mathsf{ret}.\mathsf{Q}\}) \qquad \Gamma \vdash \mathsf{C} : \mathsf{Class} \qquad \Gamma \vdash \mathsf{m} : \mathsf{Method}\end{array}}{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}[\mathsf{y}/\mathtt{this}] * \mathsf{x} \mapsto \mathsf{y}.\mathsf{m} * \mathsf{y} : \mathsf{C}\}\mathtt{fork}(\mathsf{x})\{\mathsf{emp}\}}$$

$$\frac{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P} * \mathsf{x} =_{\mathsf{Val}} \mathsf{y}\}\bar{\mathsf{s}}_1\{\mathsf{Q}\} \qquad \Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P} * \mathsf{x} \neq_{\mathsf{Val}} \mathsf{y}\}\bar{\mathsf{s}}_2\{\mathsf{Q}\}}{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}\}\mathtt{if}\ (\mathsf{x} = \mathsf{y})\ \mathtt{then}\ \bar{\mathsf{s}}_1\ \mathtt{else}\ \bar{\mathsf{s}}_2\{\mathsf{Q}\}}$$

Method verification

To verify a method, we verify the method body.

$$\frac{\begin{array}{c}\mathsf{C} \in \mathit{CName} \qquad \mathsf{m} \in \mathit{MName} \\ \mathsf{body}(\mathsf{C},\mathsf{m}) = \mathsf{C}\ \mathsf{m}(\Delta)\{\overline{\mathsf{C}\mathsf{y}};\mathsf{s};\mathtt{return}\ \mathsf{r}\} \qquad \mathsf{vars}(\Delta,\mathtt{this}) \cap \mathsf{mod}(\mathsf{s}) = \emptyset \\ \Gamma;\Delta,\mathtt{this} \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma;\Delta,\mathtt{this},\mathsf{ret} \vdash \mathsf{Q} : \mathsf{Prop} \\ \Gamma \mid \Phi \vdash (\Delta,\bar{\mathsf{y}},\mathtt{this}).\{\mathsf{P} * \mathtt{this} : \mathsf{C}\}\mathsf{s}\{\mathsf{Q}[\mathsf{r}/\mathsf{ret}]\}\end{array}}{\Gamma \mid \Phi \vdash \mathsf{C}.\mathsf{m} : (\Delta).\{\mathsf{P}\}\{\mathsf{ret}.\mathsf{Q}\}}$$

Mini C$^\sharp$ lacks constructor bodies. To allocate a phantom field we require that the given field does not already exist. Constructors are thus mainly useful for allocating phantom fields, as objects cannot have phantom fields before they have been allocated. The constructor verification rule thus allows the user to introduce an arbitrary (finite) number of phantom fields on the newly created object.

$$\frac{\begin{array}{c}\mathsf{C} \in \mathit{CName} \qquad \mathsf{fields}(\mathsf{C}) = \bar{\mathsf{f}} \qquad \Gamma;\mathtt{this} \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma;\mathtt{this},\mathsf{ret} \vdash \mathsf{Q} : \mathsf{Prop} \\ \Gamma \mid \Phi \vdash (\bar{\mathsf{y}},\mathtt{this}).\{\mathsf{P} * \mathtt{this}.\bar{\mathsf{f}} \mapsto \mathsf{null} * \overline{\mathtt{this}\_ \mapsto \_} * \mathtt{this} : \mathsf{C}\}\mathtt{skip}\{\mathsf{Q}[\mathtt{this}/\mathsf{ret}]\}\end{array}}{\Gamma \mid \Phi \vdash \mathsf{C} : \{\mathsf{P}\}\{\mathsf{ret}.\mathsf{Q}\}}$$

# 3 Interpretation

*Types* $\boxed{[\![\vdash \tau : \mathsf{Type}]\!] \in \{X \in \mathit{obj}(\mathcal{S}) \mid \mathit{total}(X) \wedge \mathit{inhabited}(X)\}}$

$$\begin{aligned}
[\![\vdash 1 : \mathsf{Type}]\!] &= 1 \\
[\![\vdash \tau \to \sigma : \mathsf{Type}]\!] &= [\![\vdash \tau : \mathsf{Type}]\!] \to [\![\vdash \sigma : \mathsf{Type}]\!] \\
[\![\vdash \tau \times \sigma : \mathsf{Type}]\!] &= [\![\vdash \tau : \mathsf{Type}]\!] \times [\![\vdash \sigma : \mathsf{Type}]\!] \\
[\![\vdash \tau + \sigma : \mathsf{Type}]\!] &= [\![\vdash \tau : \mathsf{Type}]\!] + [\![\vdash \sigma : \mathsf{Type}]\!] \\
[\![\vdash \mathcal{P}(\tau) : \mathsf{Type}]\!] &= \mathcal{P}([\![\vdash \tau : \mathsf{Type}]\!]) \\
[\![\vdash \mathsf{Prop} : \mathsf{Type}]\!] &= \mathit{Prop} \\
[\![\vdash \mathsf{Spec} : \mathsf{Type}]\!] &= \Omega \\
[\![\vdash \Delta(X) : \mathsf{Type}]\!] &= \Delta(X)
\end{aligned}$$

*Context* $\boxed{[\![\Gamma]\!], [\![\Delta]\!] \in \{X \in \mathit{obj}(\mathcal{S}) \mid \mathit{total}(X) \wedge \mathit{inhabited}(X)\}}$

$$[\![\Gamma, \mathsf{x} : \tau]\!] \stackrel{\text{def}}{=} [\![\Gamma]\!] \times [\![\vdash \tau : \mathsf{Type}]\!] \qquad [\![\varepsilon]\!] \stackrel{\text{def}}{=} 1$$

$$[\![\Delta, \mathsf{x} : \mathsf{Val}]\!] \stackrel{\text{def}}{=} [\![\Delta]\!] \times [\![\vdash \mathsf{Val} : \mathsf{Type}]\!] \qquad [\![\varepsilon]\!] \stackrel{\text{def}}{=} 1$$

*Terms* $\boxed{[\![\Gamma; \Delta \vdash \mathsf{M} : \tau]\!] : [\![\Gamma]\!] \times [\![\Delta]\!] \to [\![\tau]\!] \in \mathcal{S}}$

Lambda calculus

$$
\begin{aligned}
[\![\Gamma; \Delta \vdash \mathsf{x} : \tau]\!](\vartheta, \delta) &= \pi_{\mathsf{x}}(\vartheta) \\
[\![\Gamma; \Delta \vdash \mathsf{x} : \mathsf{Val}]\!](\vartheta, \delta) &= \pi_{\mathsf{x}}(\delta) \\
[\![\Gamma; \Delta \vdash \lambda \mathsf{x} : \tau.\ \mathsf{M} : \tau \to \sigma]\!](\vartheta, \delta) &= \lambda v \in [\![\vdash \tau : \mathsf{Type}]\!].\ [\![\Gamma, \mathsf{x} : \tau; \Delta \vdash \mathsf{M} : \sigma]\!]((\vartheta, v), \theta) \\
[\![\Gamma; \Delta \vdash \mathsf{M}\ \mathsf{N} : \sigma]\!](\vartheta, \delta) &= ([\![\Gamma; \Delta \vdash \mathsf{M} : \tau \to \sigma]\!](\vartheta, \delta))([\![\Gamma; \Delta \vdash \mathsf{N} : \tau]\!](\vartheta, \delta))
\end{aligned}
$$

Assertion logic

$$
\begin{aligned}
[\![\Gamma; \Delta \vdash \bot : \mathsf{Prop}]\!](\vartheta, \delta) &= \emptyset \\
[\![\Gamma; \Delta \vdash \top : \mathsf{Prop}]\!](\vartheta, \delta) &= \mathcal{M} \\
[\![\Gamma; \Delta \vdash \mathsf{P} \wedge \mathsf{Q} : \mathsf{Prop}]\!](\vartheta, \delta) &= [\![\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop}]\!](\vartheta, \delta) \cap [\![\Gamma; \Delta \vdash \mathsf{Q} : \mathsf{Prop}]\!](\vartheta, \delta) \\
[\![\Gamma; \Delta \vdash \mathsf{P} \vee \mathsf{Q} : \mathsf{Prop}]\!](\vartheta, \delta) &= [\![\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop}]\!](\vartheta, \delta) \cup [\![\Gamma; \Delta \vdash \mathsf{Q} : \mathsf{Prop}]\!](\vartheta, \delta) \\
[\![\Gamma; \Delta \vdash \mathsf{P} \Rightarrow \mathsf{Q} : \mathsf{Prop}]\!](\vartheta, \delta) &= \{m \in \mathcal{M} \mid \forall m \leq n. \\
&\qquad n \in [\![\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop}]\!](\vartheta, \delta)\ \Rightarrow \\
&\qquad n \in [\![\Gamma; \Delta \vdash \mathsf{Q} : \mathsf{Prop}]\!](\vartheta, \delta)\} \\
[\![\Gamma; \Delta \vdash \forall \mathsf{x} : \tau.\ \mathsf{P} : \mathsf{Prop}]\!](\vartheta, \delta) &= \bigcap_{v \in [\![\vdash \tau : \mathsf{Type}]\!]} [\![\Gamma, \mathsf{x} : \tau; \Delta \vdash \mathsf{P} : \mathsf{Prop}]\!]((\vartheta, v), \delta) \\
[\![\Gamma; \Delta \vdash \exists \mathsf{x} : \tau.\ \mathsf{P} : \mathsf{Prop}]\!](\vartheta, \delta) &= \bigcup_{v \in [\![\vdash \tau : \mathsf{Type}]\!]} [\![\Gamma, \mathsf{x} : \tau; \Delta \vdash \mathsf{P} : \mathsf{Prop}]\!]((\vartheta, v), \delta)
\end{aligned}
$$

Specification logic

$$
\begin{aligned}
[\![\Gamma \vdash \bot : \mathsf{Spec}]\!](\vartheta) &= \bot \\
[\![\Gamma \vdash \top : \mathsf{Spec}]\!](\vartheta) &= \top \\
[\![\Gamma \vdash \mathsf{S} \wedge \mathsf{T} : \mathsf{Spec}]\!](\vartheta) &= [\![\Gamma \vdash \mathsf{S} : \mathsf{Spec}]\!](\vartheta) \wedge [\![\Gamma \vdash \mathsf{T} : \mathsf{Spec}]\!](\vartheta) \\
[\![\Gamma \vdash \mathsf{S} \vee \mathsf{T} : \mathsf{Spec}]\!](\vartheta) &= [\![\Gamma \vdash \mathsf{S} : \mathsf{Spec}]\!](\vartheta) \vee [\![\Gamma \vdash \mathsf{T} : \mathsf{Spec}]\!](\vartheta) \\
[\![\Gamma \vdash \mathsf{S} \Rightarrow \mathsf{T} : \mathsf{Spec}]\!](\vartheta) &= [\![\Gamma \vdash \mathsf{S} : \mathsf{Spec}]\!](\vartheta) \Rightarrow [\![\Gamma \vdash \mathsf{T} : \mathsf{Spec}]\!](\vartheta)
\end{aligned}
$$

$$
\begin{aligned}
[\![\Gamma \vdash \forall \mathsf{x} : \tau.\ \mathsf{S} : \mathsf{Spec}]\!](\vartheta) &= \forall v \in [\![\vdash \tau : \mathsf{Type}]\!].\ [\![\Gamma, \mathsf{x} : \tau \vdash \mathsf{S} : \mathsf{Spec}]\!]((\vartheta, v)) \\
[\![\Gamma \vdash \exists \mathsf{x} : \tau.\ \mathsf{S} : \mathsf{Spec}]\!](\vartheta) &= \exists v \in [\![\vdash \tau : \mathsf{Type}]\!].\ [\![\Gamma, \mathsf{x} : \tau \vdash \mathsf{S} : \mathsf{Spec}]\!]((\vartheta, v)) \\
[\![\Gamma \vdash \mathsf{M} =_\tau \mathsf{N} : \mathsf{Spec}]\!](\vartheta) &= [\![\Gamma \vdash \mathsf{M} : \tau]\!](\vartheta) = [\![\Gamma \vdash \mathsf{N} : \tau]\!](\vartheta)
\end{aligned}
$$

Separation logic

$$
\begin{aligned}
[\![\Gamma; \Delta \vdash \mathsf{emp} : \mathsf{Prop}]\!](\vartheta, \delta) &= emp \\
[\![\Gamma; \Delta \vdash \mathsf{P} * \mathsf{Q} : \mathsf{Prop}]\!](\vartheta, \delta) &= [\![\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop}]\!](\vartheta, \delta) * [\![\Gamma; \Delta \vdash \mathsf{Q} : \mathsf{Prop}]\!](\vartheta, \delta) \\
[\![\Gamma; \Delta \vdash \mathsf{P} \mathrel{-\!\!*} \mathsf{Q} : \mathsf{Prop}]\!](\vartheta, \delta) &= [\![\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop}]\!](\vartheta, \delta) \mathrel{-\!\!*} [\![\Gamma; \Delta \vdash \mathsf{Q} : \mathsf{Prop}]\!](\vartheta, \delta)
\end{aligned}
$$

C♯

$$
\begin{aligned}
[\![\Gamma; \Delta \vdash \Delta_X(x) : \Delta(X)]\!]_n(\vartheta, \delta) &= x \\
[\![\Gamma; \Delta \vdash \mathsf{M.F} \mapsto \mathsf{N} : \mathsf{Prop}]\!](\vartheta, \delta) &= \{m \in \mathcal{M} \mid \exists o \in \mathit{OId}.\ \exists v \in \mathit{CVal}.\ \exists f \in \mathit{FName}. \\
&\qquad o = [\![\Gamma \vdash \mathsf{M} : \mathsf{Val}]\!](\vartheta, \delta) \wedge \\
&\qquad f = [\![\Gamma \vdash \mathsf{F} : \mathsf{Field}]\!](\vartheta, \delta) \wedge \\
&\qquad v = [\![\Gamma \vdash \mathsf{N} : \mathsf{Val}]\!](\vartheta, \delta) \wedge \\
&\qquad (m.l.h.o)(o, f) = v\} \\
[\![\Gamma; \Delta \vdash \mathsf{M}_\mathsf{F} \mapsto \mathsf{N} : \mathsf{Prop}]\!](\vartheta, \delta) &= \{m \in \mathcal{M} \mid \exists o \in \mathit{OId}.\ \exists v \in \mathit{CVal}.\ \exists f \in \mathit{FName}. \\
&\qquad o = [\![\Gamma \vdash \mathsf{M} : \mathsf{Val}]\!](\vartheta, \delta) \wedge \\
&\qquad f = [\![\Gamma \vdash \mathsf{F} : \mathsf{Field}]\!](\vartheta, \delta) \wedge \\
&\qquad v = [\![\Gamma \vdash \mathsf{N} : \mathsf{Val}]\!](\vartheta, \delta) \wedge \\
&\qquad (m.l.p)(o, f) = v\} \\
[\![\Gamma; \Delta \vdash \mathsf{N}_1 \mapsto \mathsf{N}_2.\mathsf{M} : \mathsf{Prop}]\!](\vartheta, \delta) &= \{m \in \mathcal{M} \mid \exists c \in \mathit{CId}.\ \exists o \in \mathit{OId}.\ \exists m \in \mathit{MName}. \\
&\qquad c = [\![\Gamma \vdash \mathsf{N}_1 : \mathsf{Val}]\!](\vartheta, \delta) \wedge \\
&\qquad o = [\![\Gamma \vdash \mathsf{N}_2 : \mathsf{Val}]\!](\vartheta, \delta) \wedge \\
&\qquad m = [\![\Gamma \vdash \mathsf{M} : \mathsf{Method}]\!](\vartheta, \delta) \wedge \\
&\qquad \pi_c(\pi_h(\pi_l(m)))(c) = (o, m)\} \\
[\![\Gamma; \Delta \vdash \mathsf{M} : \mathsf{C} : \mathsf{Prop}]\!](\vartheta, \delta) &= \{m \in \mathcal{M} \mid \\
&\qquad (m.l.h.t)([\![\Gamma \vdash \mathsf{M} : \mathsf{Val}]\!](\vartheta, \delta)) = [\![\Gamma \vdash \mathsf{C} : \mathsf{Class}]\!](\vartheta, \delta)\}
\end{aligned}
$$

Embeddings and guarded recursion

$$
\begin{aligned}
[\![\Gamma \vdash \mathsf{valid(P)} : \mathsf{Spec}]\!](\vartheta) &= \mathit{valid}([\![\Gamma; - \vdash \mathsf{P} : \mathsf{Prop}]\!](\vartheta)) \\
[\![\Gamma \vdash \mathsf{guarded}_\tau(\mathsf{M}) : \mathsf{Spec}]\!](\vartheta) &= \mathit{guarded}([\![\Gamma \vdash \mathsf{M} : (\tau \to \mathsf{Prop}) \to (\tau \to \mathsf{Prop})]\!](\vartheta)) \\
[\![\Gamma \vdash \triangleright \mathsf{S} : \mathsf{Spec}]\!](\vartheta) &= \triangleright([\![\Gamma \vdash \mathsf{S} : \mathsf{Spec}]\!](\vartheta))
\end{aligned}
$$

$$
\begin{aligned}
[\![\Gamma; \Delta \vdash \mathsf{spec(S)} : \mathsf{Prop}]\!](\vartheta, \delta) &= \mathit{spec}([\![\Gamma \vdash \mathsf{S} : \mathsf{Spec}]\!](\vartheta)) \\
[\![\Gamma; \Delta \vdash \mathsf{fix}_\tau(\mathsf{M}) : \tau \to \mathsf{Prop}]\!](\vartheta, \delta) &= \mathit{fix}([\![\Gamma; \Delta \vdash \mathsf{M} : (\tau \to \mathsf{Prop}) \to (\tau \to \mathsf{Prop})]\!](\vartheta, \delta)) \\
[\![\Gamma; \Delta \vdash \triangleright \mathsf{P} : \mathsf{Prop}]\!](\vartheta, \delta) &= \triangleright([\![\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop}]\!](\vartheta, \delta))
\end{aligned}
$$

Concurrent abstract predicates

$$
\begin{aligned}
&[\![\Gamma; \Delta \vdash \mathsf{region(M, N, R)} : \mathsf{Prop}]\!](\vartheta, \delta) = \\
&\quad \{m \in \mathcal{M} \mid \exists M \in \mathcal{P}(\Delta(SId)).\ \exists T : \Delta(AId \to \mathcal{P}(SId \times SId)).\ \exists R \in \Delta(RId). \\
&\qquad M = [\![\Gamma; \Delta \vdash \mathsf{M} : \mathcal{P}(\mathsf{SId})]\!](\vartheta, \delta) \wedge \\
&\qquad \phi(T) = [\![\Gamma; \Delta \vdash \mathsf{T} : \mathsf{Action} \to \mathcal{P}(\mathsf{SId} \times \mathsf{SId})]\!](\vartheta, \delta) \wedge \\
&\qquad R = [\![\Gamma; \Delta \vdash \mathsf{R} : \mathsf{Region}]\!](\vartheta, \delta) \wedge \\
&\qquad m \in region(M, T, R)\}
\end{aligned}
$$

$$
[\![\Gamma; \Delta \vdash \mathsf{rintr(I, R)} : \mathsf{Prop}]\!](\vartheta, \delta) = rintr([\![\Gamma; \Delta \vdash \mathsf{I} : \mathsf{SId} \to \mathsf{Prop}]\!](\vartheta, \delta), [\![\Gamma; \Delta \vdash \mathsf{R} : \mathsf{Region}]\!](\vartheta, \delta))
$$

$$
\begin{aligned}
[\![\Gamma; \Delta \vdash [\mathsf{A}]^{\mathsf{R}}_{\mathsf{P}} : \mathsf{Prop}]\!](\vartheta, \delta) = \ & action([\![\Gamma; \Delta \vdash \mathsf{A} : \mathsf{Action}]\!](\vartheta, \delta), \\
& [\![\Gamma; \Delta \vdash \mathsf{R} : \mathsf{Region}]\!](\vartheta, \delta), \\
& [\![\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Perm}]\!](\vartheta, \delta))
\end{aligned}
$$

where $\phi$ is defined in Lemma 60.

$$
[\![\Gamma \vdash \mathsf{stable(P)} : \mathsf{Spec}]\!](\vartheta) = stable([\![\Gamma \vdash \mathsf{P} : \mathsf{Prop}]\!](\vartheta))
$$

$$
[\![\Gamma \vdash \mathsf{P} \sqsubseteq^{\mathsf{R}} \mathsf{Q} : \mathsf{Spec}]\!](\vartheta) = [\![\Gamma \vdash \mathsf{P} : \mathsf{Prop}]\!](\vartheta) \sqsubseteq_{[\![\Gamma \vdash \mathsf{R} : \mathcal{P}(\mathsf{Region})]\!](\vartheta)} [\![\Gamma \vdash \mathsf{Q} : \mathsf{Prop}]\!](\vartheta)
$$

Hoare assertions

$$
\begin{array}{l}
[\![\Gamma \vdash (\Delta).\{\mathsf{P}\}\bar{\mathsf{s}}\{\mathsf{Q}\}]\!](\vartheta) = \\
\quad \exists p, q \in [\![\Delta]\!] \to \mathit{Prop}.\ \forall t \in \Delta(\mathit{TId}).\ \forall l \in \Delta(\mathit{Stack}). \\
\qquad p = \lambda\delta \in [\![\Delta]\!].\ [\![\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop}]\!](\vartheta, \delta) \wedge \\
\qquad q = \lambda\delta \in [\![\Delta]\!].\ [\![\Gamma; \Delta \vdash \mathsf{Q} : \mathsf{Prop}]\!](\vartheta, \delta) \wedge \\
\qquad \mathit{safe}((t, l, \mathit{stm}(\bar{\mathsf{s}})), ||p||_\Delta(l), ||q||_\Delta) \\
[\![\Gamma \vdash \mathsf{C.M} : (\Delta).\{\mathsf{P}\}\{\mathsf{r.Q}\}]\!](\vartheta) = \\
\quad \exists p \in [\![\Delta, \mathtt{this}]\!] \to \mathit{Prop}.\ \exists q \in [\![\Delta, \mathtt{this}, \mathsf{r}]\!]. \\
\quad \exists c \in \Delta(\mathit{CName}).\ \exists m \in \Delta(\mathit{MName}). \\
\quad \forall t \in \Delta(\mathit{TId}).\ \forall l \in \Delta(\mathit{Stack}).\ \forall o_t \in \Delta(\mathit{OId}).\ \forall v_{\bar{\mathsf{y}}} \in \Delta(\mathit{CVal}). \\
\qquad p = \lambda\delta \in [\![\Delta, \mathtt{this}]\!].\ [\![\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop}]\!](\vartheta, \delta) \wedge \\
\qquad q = \lambda\delta \in [\![\Delta, \mathtt{this}, \mathsf{z}]\!].\ [\![\Gamma; \Delta \vdash \mathsf{Q}[\mathsf{z}/\mathsf{r}] : \mathsf{Prop}]\!](\vartheta, \delta) \wedge \\
\qquad c = [\![\Gamma; \Delta \vdash \mathsf{C} : \mathsf{Class}]\!](\vartheta) \wedge \\
\qquad m = [\![\Gamma; \Delta \vdash \mathsf{M} : \mathsf{Method}]\!](\vartheta) \wedge \\
\qquad \mathsf{body}(c, m) = \{\overline{\mathsf{Cy}}; \bar{\mathsf{s}}; \mathtt{return\ z}\} \wedge \\
\qquad \mathit{safe}((t, (l[\mathtt{this} \mapsto o_t, \bar{\mathsf{y}} \mapsto v_{\bar{\mathsf{y}}}]), \mathit{stm}(\bar{\mathsf{s}})), ||p||_{\Delta,\mathtt{this}}(l[\mathtt{this} \mapsto o_t]), ||q||_{\Delta,\mathtt{this},\mathsf{z}}) \\
[\![\Gamma \vdash (\Delta).\langle\mathsf{P}\rangle s \langle\mathsf{Q}\rangle^{\mathsf{R}} : \mathsf{Spec}]\!](\vartheta) = \\
\quad \exists p, q \in [\![\Delta]\!] \to \mathit{Prop}.\ \exists rt \in \mathcal{P}(\Delta(\mathit{RId})).\ \forall t \in \Delta(\mathit{TId}).\ \forall l, l' \in \Delta(\mathit{Stack}).\ \forall a \in \Delta(\mathit{Act}). \\
\qquad p = \lambda\delta \in [\![\Delta]\!].\ [\![\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop}]\!](\vartheta, \delta) \wedge \\
\qquad q = \lambda\delta \in [\![\Delta]\!].\ [\![\Gamma; \Delta \vdash \mathsf{Q} : \mathsf{Prop}]\!](\vartheta, \delta) \wedge \\
\qquad rt = [\![\Gamma \vdash \mathsf{R} : \mathcal{P}(\mathsf{Region})]\!](\vartheta) \wedge \\
\qquad (l, s) \xrightarrow{a} (l', \varepsilon) \wedge \\
\qquad a\ \mathrm{sat}^{rt}\ \{||p||_\Delta(l)\}\{||q||_\Delta(l')\}
\end{array}
$$

where $||-||_\Delta : ([\![\Delta]\!] \to \mathit{Prop}) \to (\Delta(\mathit{Stack}) \to \mathit{Prop})$ is defined as follows:

$$||p||_\Delta(l) \stackrel{\text{def}}{=} \begin{cases} p(l(\mathsf{x}_1), ..., l(\mathsf{x}_n)) & \text{if } \Delta = \mathsf{x}_1, ..., \mathsf{x}_n \text{ and } \mathsf{x}_1, ..., \mathsf{x}_n \in \mathit{dom}(l) \\ \emptyset & \text{otherwise} \end{cases}$$

*Assertion logic entailment* $\boxed{[\![\Gamma; \Delta \mid \Phi \mid \mathsf{P} \vdash \mathsf{Q}]\!] : \Omega}$

$$
\begin{array}{l}
[\![\Gamma; \Delta \mid \Phi \mid \mathsf{P} \vdash \mathsf{Q}]\!] = \\
\quad \forall \vartheta \in [\![\Gamma]\!].\ \forall \delta \in [\![\Delta]\!].\ [\![\Phi]\!](\vartheta)\ \Rightarrow \\
\qquad \forall m \in \mathcal{M}.\ m \in [\![\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop}]\!](\vartheta, \delta) \Rightarrow m \in [\![\Gamma; \Delta \vdash \mathsf{Q} : \mathsf{Prop}]\!](\vartheta, \delta)
\end{array}
$$

*Specification logic entailment* $\boxed{[\![\Gamma \mid \mathsf{S}_1, ..., \mathsf{S}_n \vdash \mathsf{T}]\!] : \Omega}$

$$[\![\Gamma \mid \mathsf{S}_1, ..., \mathsf{S}_n \vdash \mathsf{T}]\!] = \forall \vartheta \in [\![\Gamma]\!]. \left( \bigwedge_{i \in \{1,...,n\}} [\![\Gamma \vdash \mathsf{S}_i : \mathsf{Spec}]\!](\vartheta) \right) \Rightarrow [\![\Gamma \vdash \mathsf{T} : \mathsf{Spec}]\!](\vartheta)$$

## 4 Refinement and Fine-Grained Concurrency

In this section we provide further details about the verification of a fine-grained concurrent bag, implemented using cooperation, against a refinable specification. The implementation uses a side-channel to allow a pusher and a popper to synchronize and exchange an element, with touching the main data structure – thus reducing contention. Please see the accompanying article for an explanation of the specification and overall motivation.

Specification.

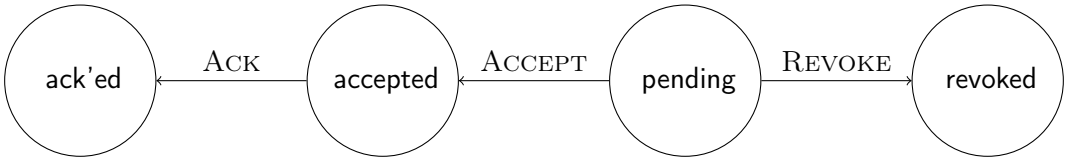
$$
\begin{array}{l}
\exists \mathsf{bag} : \mathsf{RId} \times \mathsf{Val} \to \mathsf{Prop}. \\
\quad \{\mathsf{emp}\} \\
\qquad \mathbf{new}\ \mathsf{Bag}(-) \\
\quad \left\{ \mathsf{ret}.\ \exists \mathsf{n} : \mathsf{RId}.\ \mathsf{bag}(\mathsf{n}, \mathsf{ret}) * \mathsf{ret}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \emptyset \right\}
\end{array}
$$

$$
\begin{array}{l}
\forall \mathsf{P}, \mathsf{Q} : \mathsf{Val} \times \mathsf{Val} \to \mathsf{Prop}.\ \forall \mathsf{n} : \mathsf{RId}. \\
\quad (\forall \mathsf{x}, \mathsf{y} : \mathsf{Val}.\ \mathsf{stable}(\mathsf{P}(\mathsf{x}, \mathsf{y})) \wedge \mathsf{stable}(\mathsf{Q}(\mathsf{x}, \mathsf{y})) \wedge \\
\quad (\forall \mathsf{X} : \mathcal{P}_m(\mathsf{Val}).\ \forall \mathsf{x}, \mathsf{y} : \mathsf{Val}.\ \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \mathsf{X} * \mathsf{P}(\mathsf{x}, \mathsf{y}) \sqsubseteq^{RId \setminus \mathsf{n}^*} \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} (\mathsf{X} \cup \{\mathsf{y}\}) * \mathsf{Q}(\mathsf{x}, \mathsf{y})) \ \Rightarrow \\
\qquad \{\mathsf{bag}(\mathsf{n}, \mathsf{x}) * \mathsf{P}(\mathsf{x}, \mathsf{y})\} \\
\qquad\quad \mathsf{x}.\mathsf{Push}(\mathsf{y}) \\
\qquad \{\mathsf{bag}(\mathsf{n}, \mathsf{x}) * \mathsf{Q}(\mathsf{x}, \mathsf{y})\}
\end{array}
$$

$$
\begin{array}{l}
\forall \mathsf{P} : \mathsf{Val} \to \mathsf{Prop}.\ \forall \mathsf{Q} : \mathsf{Val} \times \mathsf{Val} \to \mathsf{Prop}.\ \forall \mathsf{n} : \mathsf{RId}. \\
\quad (\forall \mathsf{x}, \mathsf{y} : \mathsf{Val}.\ \mathsf{stable}(\mathsf{P}(\mathsf{x})) \wedge \mathsf{stable}(\mathsf{Q}(\mathsf{x}, \mathsf{y})) \wedge \\
\quad (\forall \mathsf{X} : \mathcal{P}_m(\mathsf{Val}).\ \forall \mathsf{x}, \mathsf{y} : \mathsf{Val}.\ \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} (\{\mathsf{y}\} \cup \mathsf{X}) * \mathsf{P}(\mathsf{x}) \sqsubseteq^{RId \setminus \mathsf{n}^*} \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \mathsf{X} * \mathsf{Q}(\mathsf{x}, \mathsf{y})) \wedge \\
\quad (\forall \mathsf{x} : \mathsf{Val}.\ \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \emptyset * \mathsf{P}(\mathsf{x}) \sqsubseteq^{RId \setminus \mathsf{n}^*} \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \emptyset * \mathsf{Q}(\mathsf{x}, \mathsf{null})) \ \Rightarrow \\
\qquad \{\mathsf{bag}(\mathsf{n}, \mathsf{x}) * \mathsf{P}(\mathsf{x})\} \\
\qquad\quad \mathsf{x}.\mathsf{Pop}(-) \\
\qquad \{\mathsf{ret}.\ \mathsf{bag}(\mathsf{n}, \mathsf{x}) * \mathsf{Q}(\mathsf{x}, \mathsf{ret})\}
\end{array}
$$

where $\mathsf{n}^*$ is shorthand for $\{\mathsf{m} : \mathsf{RId} \mid \mathsf{n} \leq \mathsf{m}\}$.

Predicate Definitions. To verify the implementation against the above specification we first need to provide concrete instantiations of the abstract bag representation predicate. The main difficulty in defining the bag predicate is the protocol governing the side-channel. Offers on this side-channel are governed by the following labelled transition system, $\mathsf{T}_{offer}$:




An offer starts in the pending state. From this state any popper can accept the offer or the pusher that made the offer can revoke it. Finally, from the accepted state, the pusher that made the offer can acknowledge that the offer has been accepted. To reason about offers we introduce two representation predicates, isOffer and myOffer, to assert existence and ownership of a given offer, respectively:

$$\begin{aligned}\mathsf{isOffer}(\mathsf{n},\mathsf{b},\mathsf{x}) \stackrel{\text{def}}{=}\ & \exists \mathsf{P},\mathsf{Q} : \mathsf{Val}\times\mathsf{Val}\to\mathsf{Prop}.\ \exists \mathsf{y}:\mathsf{Val}.\ \exists \mathsf{m}:\mathsf{RId}.\\ & \mathsf{region}(\{\mathsf{ack'ed},\mathsf{accepted},\mathsf{pending},\mathsf{revoked}\},\mathsf{T}_{\mathit{offer}},\mathsf{n}) * \mathsf{n}<\mathsf{m} *\\ & \mathsf{rintr}(\mathsf{I}_{\mathit{offer}}(\mathsf{n},\mathsf{P},\mathsf{Q},\mathsf{b},\mathsf{x},\mathsf{y}),\mathsf{m}) * [\textsc{Accept}]^{\mathsf{m}}_{\_} * \mathsf{x.value} \Rightarrow \mathsf{y}\end{aligned}$$

$$\begin{aligned}\mathsf{myOffer}(\mathsf{n},\mathsf{P},\mathsf{Q},\mathsf{b},\mathsf{x},\mathsf{y}) \stackrel{\text{def}}{=}\ & \exists \mathsf{m}:\mathsf{RId}.\ \mathsf{n}<\mathsf{m} *\\ & \mathsf{region}(\{\mathsf{accepted},\mathsf{pending}\},\mathsf{T}_{\mathit{offer}},\mathsf{m}) *\\ & \mathsf{rintr}(\mathsf{I}_{\mathit{offer}}(\mathsf{n},\mathsf{P},\mathsf{Q},\mathsf{b},\mathsf{x},\mathsf{y}),\mathsf{m}) * \mathsf{x.value} \Rightarrow \mathsf{y} *\\ & [\textsc{Revoke}]^{\mathsf{m}}_1 * [\textsc{Ack}]^{\mathsf{m}}_1\end{aligned}$$

where

$$\mathsf{I}_{\mathit{offer}}(\mathsf{n},\mathsf{P},\mathsf{Q},\mathsf{b},\mathsf{x},\mathsf{y})(\mathsf{pending}) \stackrel{\text{def}}{=} \mathsf{x.state}\mapsto 0 * \mathsf{P}(\mathsf{b},\mathsf{y}) *$$
$$\mathsf{spec}(\forall \mathsf{X}:\mathcal{P}_m(\mathsf{Val}).\ \forall \mathsf{x},\mathsf{y}:\mathsf{Val}.\ \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \mathsf{X} * \mathsf{P}(\mathsf{x},\mathsf{y}) \sqsubseteq^{\mathit{RId}\setminus \mathsf{n}^*} \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} (\mathsf{X}\cup\{\mathsf{y}\}) * \mathsf{Q}(\mathsf{x},\mathsf{y}))$$

and

$$\mathsf{I}_{\mathit{offer}}(\mathsf{n},\mathsf{P},\mathsf{Q},\mathsf{b},\mathsf{x},\mathsf{y})(\mathsf{accepted}) \stackrel{\text{def}}{=} \mathsf{x.state}\mapsto 1 * \mathsf{Q}(\mathsf{b},\mathsf{y})$$
$$\mathsf{I}_{\mathit{offer}}(\mathsf{n},\mathsf{P},\mathsf{Q},\mathsf{b},\mathsf{x},\mathsf{y})(\mathsf{revoked}) \stackrel{\text{def}}{=} \mathsf{x.state}\mapsto 2$$
$$\mathsf{I}_{\mathit{offer}}(\mathsf{n},\mathsf{P},\mathsf{Q},\mathsf{b},\mathsf{x},\mathsf{y})(\mathsf{ack'ed}) \stackrel{\text{def}}{=} \mathsf{x.state}\mapsto 1$$

The isOffer assertion asserts non-exclusive ownership of the Accept action and is thus freely duplicable. The myOffer assertion asserts exclusive ownership of the Revoke and Ack actions and is thus not duplicable.

We can now define the bag assertion as follows:

$$\mathsf{bag}(\mathsf{n},\mathsf{x}) \stackrel{\text{def}}{=} \mathsf{region}(\{\mathsf{bag}\},\mathsf{T}_{\mathit{bag}},\mathsf{n}) * \mathsf{rintr}(\mathsf{I}_{\mathit{bag}}(\mathsf{n},\mathsf{x}),\mathsf{n})$$

where

$$\begin{aligned}\mathsf{I}_{\mathit{bag}}(\mathsf{n},\mathsf{x})(\mathsf{bag}) \stackrel{\text{def}}{=}\ & \exists \mathsf{y},\mathsf{z}:\mathsf{Val}.\ \exists l : \mathit{seq}\ \mathsf{Val}.\\ & \mathsf{x.head}\mapsto \mathsf{y} * \mathsf{x.offer}\mapsto \mathsf{z} * \mathsf{lst}_r(\mathsf{y},l) *\\ & \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \mathit{mem}(l) * (\mathsf{z}=\mathsf{null}\vee \mathsf{isOffer}(\mathsf{n},\mathsf{x},\mathsf{z}))\end{aligned}$$

and $\mathsf{T}_{\mathit{bag}}$ denotes the following labelled transition system:


Stability. The isOffer assertion is trivially stable as it asserts that the region may be in any of the possible offer region states. The myOffer assertion is stable because it asserts exclusive ownership of the REVOKE and ACT actions. Finally, the bag assertion is trivially stable as there is only one bag state.

Proof Sketch.

```
internal class Node⟨A⟩ {
  internal Node⟨A⟩ next;
  internal A value;

  public Node(A value) {
{this.next ↦ null ∗ this.value ↦ null}
    this.value = value;
{this.next ↦ null ∗ this.value ↦ value}
  }
}

internal class Offer⟨A⟩ {
  internal int state;
  internal A value;

  public Offer(A value) {
{this.value ↦ null ∗ this.state ↦ 0 ∗ P(x, value)}
    this.value = value;
{this.value ↦ value ∗ this.state ↦ 0 ∗ P(x, value)}
{isOffer(n, x, this) ∗ myOffer(n, P, Q, x, this, value)}
  }

  public bool Revoke() {
{myOffer(n, P, Q, x, this, y)}
    return Interlocked.CompareExchange⟨int⟩(ref this.state, 2, 0);
{ret. (ret = true ∗ P(x, y)) ∨ (ret = false ∗ Q(x, y))}
  }
}

public class Bag⟨A⟩ where A : class {
  internal Node⟨A⟩ head;
```

internal A offer;

**public** Bag⟨A⟩() {
$\{\texttt{this.head} \mapsto \mathsf{null} * \texttt{this.offer} \mapsto \mathsf{null}\}$
$\{\texttt{this.head} \mapsto \mathsf{null} * \texttt{this.offer} \mapsto \mathsf{null} * \texttt{this}_{\mathsf{cont}} \mapsto \emptyset * \mathsf{lst}_r(\mathsf{null}, \varepsilon)\}$
$\{\texttt{this.head} \mapsto \mathsf{null} * \texttt{this.offer} \mapsto \mathsf{null} * \texttt{this}_{\mathsf{cont}} \mapsto \emptyset * \mathsf{lst}_r(\mathsf{null}, \varepsilon)\}$
$\{\exists \mathsf{n} : \mathsf{RId}.\ \mathsf{region}(\{\mathsf{bag}\}, \mathsf{T}_{bag}, \mathsf{I}_{bag}(\mathsf{m}, \texttt{this}), \mathsf{n}) * \texttt{this}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \emptyset\}$
$\{\exists \mathsf{n} : \mathsf{RId}.\ \mathsf{bag}(\mathsf{n}, \texttt{this}) * \texttt{this}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \emptyset\}$
}

In the constructor we create a new bag region in the no offer state. We thus have to show that the interpretations of the abstract bag states are stable and transfer the resources described by $\mathsf{I}_{bag}$ in the bag state to the shared region:

$$\forall \mathsf{n} : \mathsf{RId}.\ \ \texttt{this.head} \mapsto \mathsf{null} * \texttt{this.offer} \mapsto \mathsf{null} * \texttt{this}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \emptyset * \mathsf{lst}_r(\mathsf{null}, \varepsilon) \sqsubseteq \triangleright \mathsf{I}_{bag}(\mathsf{n}, \texttt{this})(\mathsf{bag})$$

This is easily seen to hold by monotonicity and the definition of $\mathsf{I}_{bag}$. Stability follows easily from the stability of isOffer.

**public void** Push(A y) {
$\{\mathsf{bag}(\mathsf{n}, \texttt{this}) * \mathsf{P}(\texttt{this}, \mathsf{y})\}$
Node⟨A⟩ nHead;
Node⟨A⟩ oHead;
Node⟨A⟩ tmp1;
Node⟨A⟩ tmp2;
Offer⟨A⟩ off;
bool done = false;
$\{\mathsf{bag}(\mathsf{n}, \texttt{this}) * \mathsf{P}(\texttt{this}, \mathsf{y}) * \mathsf{done} = \mathsf{false}\}$
nHead = **new** Node⟨A⟩(y);
$\{\mathsf{bag}(\mathsf{n}, \texttt{this}) * \mathsf{P}(\texttt{this}, \mathsf{y}) * \mathsf{done} = \mathsf{false} * \mathsf{nHead.next} \mapsto \mathsf{null} * \mathsf{nHead.value} \mapsto \mathsf{y}\}$
$\{\mathsf{bag}(\mathsf{n}, \texttt{this}) * ((\mathsf{done} = \mathsf{false} * \mathsf{P}(\texttt{this}, \mathsf{y}) * \mathsf{nHead.next} \mapsto \_ * \mathsf{nHead.value} \mapsto \mathsf{y})$
$\vee\ (\mathsf{done} = \mathsf{true} * \mathsf{Q}(\texttt{this}, \mathsf{y})))\}$
while (!done) {
$\{\mathsf{bag}(\mathsf{n}, \texttt{this}) * \mathsf{P}(\texttt{this}, \mathsf{y}) * \mathsf{done} = \mathsf{false} * \mathsf{nHead.next} \mapsto \mathsf{null} * \mathsf{nHead.value} \mapsto \mathsf{y}\}$
oHead = head;
$\{\mathsf{bag}(\mathsf{n}, \texttt{this}) * \mathsf{P}(\texttt{this}, \mathsf{y}) * \mathsf{done} = \mathsf{false} * \mathsf{nHead.next} \mapsto \mathsf{null} * \mathsf{nHead.value} \mapsto \mathsf{y} * \exists \alpha.\ \mathsf{lst}_r(\mathsf{oHead}, \alpha)\}$
nHead.next = oHead;
$\{\mathsf{bag}(\mathsf{n}, \texttt{this}) * \mathsf{P}(\texttt{this}, \mathsf{y}) * \mathsf{done} = \mathsf{false} * \mathsf{nHead.next} \mapsto \mathsf{oHead} * \mathsf{nHead.value} \mapsto \mathsf{y} * \exists \alpha.\ \mathsf{lst}_r(\mathsf{oHead}, \alpha)\}$
tmp1 = Interlocked.CompareExchange⟨Node⟨A⟩⟩(ref head, nHead, oHead);
$\{\mathsf{bag}(\mathsf{n}, \texttt{this}) * \mathsf{done} = \mathsf{false} * ((\mathsf{tmp1} = \mathsf{oHead} * \mathsf{Q}(\texttt{this}, \mathsf{y}))$
$\vee\ (\mathsf{tmp1} \neq \mathsf{oHead} * \mathsf{P}(\texttt{this}, \mathsf{y}) * \mathsf{nHead.next} \mapsto \mathsf{oHead} * \mathsf{nHead.value} \mapsto \mathsf{y} * \exists \alpha.\ \mathsf{lst}_r(\mathsf{oHead}, \alpha)))\}$
**if** (tmp1 == oHead) {
$\{\mathsf{bag}(\mathsf{n}, \texttt{this}) * \mathsf{done} = \mathsf{false} * \mathsf{tmp1} = \mathsf{oHead} * \mathsf{Q}(\texttt{this}, \mathsf{y})\}$

```
        done = true;
{bag(n, this) * ((done = false * P(this, y) * nHead.next ↦ _ * nHead.value ↦ y)
    ∨ (done = true * Q(this, y)))}
      } else {
{bag(n, this) * done = false * tmp1 ≠ oHead * P(this, y)
   * nHead.next ↦ oHead * nHead.value ↦ y * ∃α. lst_r(oHead, α)}
{bag(n, this) * done = false * tmp1 ≠ oHead * P(this, y)
   * nHead.next ↦ oHead * nHead.value ↦ y}
        off = new Offer⟨A⟩(y);
{bag(n, this) * done = false * tmp1 ≠ oHead * isOffer(n, this, off)
   * myOffer(n, P, Q, this, off, y) * nHead.next ↦ oHead * nHead.value ↦ y}
        Interlocked.CompareExchange⟨Offer⟨A⟩⟩(ref offer, off, null);
{bag(n, this) * done = false * tmp1 ≠ oHead
   * myOffer(n, P, Q, this, off, y) * nHead.next ↦ oHead * nHead.value ↦ y}
        this.offer = null;
        if (off.Revoke()) {
{bag(n, this) * done = false * tmp1 ≠ oHead
   * P(this, y) * nHead.next ↦ oHead * nHead.value ↦ y}
        } else {
{bag(n, this) * done = false * tmp1 ≠ oHead
   * Q(this, y) * nHead.next ↦ oHead * nHead.value ↦ y}
          done = true;
        }
      }
    }
  }

  public A Pop() {
{bag(n, this) * P1(this)}
    Node⟨A⟩ oHead; Node⟨A⟩ nHead; Node⟨A⟩ tmp;
    Offer⟨A⟩ off; A val;
    bool done = false;
{bag(n, this) * ((done = true * Q1(this, val)) ∨ (done = false * P1(this)))}
    while(!done) {
{bag(n, this) * done = false * P1(this)}
      oHead = this.head;
{bag(n, this) * done = false
   * ((oHead = null * Q1(this, null)) ∨ (oHead ≠ null * ∃α. lst_r(oHead, α) * P1(this))}
      if (oHead == null) {
{bag(n, this) * done = false * Q1(this, null)}
        val = null;
        done = true;
{bag(n, this) * ((done = true * Q1(this, val)) ∨ (done = false * P1(this)))}
      } else {
```

```
{bag(n, this) ∗ done = false ∗ oHead ≠ null ∗ ∃α. lst_r(oHead, α) ∗ P_1(this)}
{bag(n, this) ∗ done = false ∗ ∃x, y, α. oHead.next ↦ x ∗ oHead.value ↦ y ∗ lst_r(x, α) ∗ P_1(this)}
        nHead = oHead.next;
{bag(n, this) ∗ done = false ∗ ∃y, α. oHead.next ↦ nHead ∗ oHead.value ↦ y ∗ lst_r(nHead, α) ∗ P_1(this)}
        tmp = Interlocked.CompareExchange⟨Node⟨A⟩⟩(ref head, nHead, oHead);
{bag(n, this) ∗ done = false
  ∗ ((tmp = oHead ∗ ∃y. Q_1(this, y) ∗ oHead.value ↦ y) ∨ (tmp ≠ oHead ∗ P_1(this)))}
        if(tmp == oHead) {
{bag(n, this) ∗ done = false ∗ ∃y. Q_1(this, y) ∗ oHead.value ↦ y}
          val = oHead.value;
          done = true;
{bag(n, this) ∗ ((done = true ∗ Q_1(this, val)) ∨ (done = false ∗ P_1(this)))}
        } else {
{bag(n, this) ∗ done = false ∗ P_1(this)}
          off = this.offer;
{bag(n, this) ∗ done = false ∗ P_1(this) ∗ (off = null ∨ isOffer(n, this, off))}
          if(off != null) {
{bag(n, this) ∗ done = false ∗ P_1(this) ∗ isOffer(n, this, off)}
            tmp2 = Interlocked.CompareExchange⟨int⟩(ref off.state, 1, 0);
{bag(n, this) ∗ done = false
  ∗ ((tmp2 = 0 ∗ ∃y. Q_1(this, y) ∗ off.value ↦ y) ∨ (tmp2 ≠ 0 ∗ P_1(this)))}
            if(tmp2 == 0) {
{bag(n, this) ∗ done = false ∗ ∃y. Q_1(this, y) ∗ off.value ↦ y}
              val = off.value;
              done = true;
{bag(n, this) ∗ ((done = true ∗ Q_1(this, val)) ∨ (done = false ∗ P_1(this)))}
            } else {
{bag(n, this) ∗ done = false ∗ P_1(this)}
{bag(n, this) ∗ ((done = true ∗ Q_1(this, val)) ∨ (done = false ∗ P_1(this)))}
            }
          } else {
{bag(n, this) ∗ done = false ∗ P_1(this)}
{bag(n, this) ∗ ((done = true ∗ Q_1(this, val)) ∨ (done = false ∗ P_1(this)))}
          }
        }
      }
    }
{bag(n, this) ∗ Q_1(this, val)}
    return val;
{ret. bag(n, this) ∗ Q_1(this, ret)}
  }
}
```

Atomic updates. The most interesting atomic update is the atomic "accept offer" update in the bag pop method. If the client popping an element is successful in accepting the offer that client has to atomically perform both the view-shift provided by the push client that made the offer, and its own view-shift. This requires opening two regions: 1) the bag region containing the phantom field that stores the abstract state and 2) the offer region containing the view-shift provided by the push client that made the offer.

First we open the bag region:

$$
\begin{array}{l}
\{\mathsf{bag}(\mathsf{n},\mathtt{this}) * \mathsf{done} = \mathsf{false} * \mathsf{P}_1(\mathtt{this}) * \mathsf{isOffer}(\mathsf{n},\mathtt{this},\mathsf{off})\} \\
\quad \{\mathsf{region}(\{\mathsf{bag}\}, \mathsf{T}_{\mathit{bag}}, \mathsf{n}) * \mathsf{rintr}(\mathsf{I}_{\mathit{bag}}(\mathsf{n},\mathtt{this}), \mathsf{n}) * \mathsf{P}_1(\mathtt{this}) * \mathsf{isOffer}(\mathsf{n},\mathtt{this},\mathsf{off})\} \\
\quad \langle \triangleright \mathsf{I}_{\mathit{bag}}(\mathsf{n},\mathtt{this})(\mathsf{bag}) * \mathsf{P}_1(\mathtt{this}) * \mathsf{isOffer}(\mathsf{n},\mathtt{this},\mathsf{off}) \rangle \\
\quad \langle \triangleright \mathsf{I}_{\mathit{bag}}(\mathsf{n},\mathtt{this})(\mathsf{bag}) * \mathsf{P}_1(\mathtt{this}) * \exists \mathsf{P}_2, \mathsf{Q}_2.\ \exists \mathsf{y}.\ \exists \mathsf{m}.\ \mathsf{n} < \mathsf{m} * \\
\qquad * \mathsf{region}(\{\mathsf{ack'ed}, \mathsf{accepted}, \mathsf{pending}, \mathsf{revoked}\}, \mathsf{T}_{\mathit{offer}}, \mathsf{m}) \\
\qquad * \mathsf{rintr}(\mathsf{I}_{\mathit{offer}}(\mathsf{n}, \mathsf{P}_2, \mathsf{Q}_2, \mathtt{this}, \mathsf{off}, \mathsf{y}), \mathsf{m}) * [\textsc{Accept}]^{\mathsf{m}} * \mathsf{off.value} \mapsto \mathsf{y} \rangle \\
\qquad\qquad \mathsf{tmp2} = \mathsf{Interlocked.CompareExchange}\langle \mathbf{int} \rangle(\mathsf{ref\ off.state},\ 1,\ 0); \\
\quad \langle \triangleright \mathsf{I}_{\mathit{bag}}(\mathsf{n},\mathtt{this})(\mathsf{bag}) * ((\mathsf{tmp2} = 0 * \exists \mathsf{y}.\ \mathsf{Q}_1(\mathtt{this}, \mathsf{y}) * \mathsf{off.value} \mapsto \mathsf{y}) \vee (\mathsf{tmp2} \neq 0 * \mathsf{P}_1(\mathtt{this})))\rangle^{\mathit{RId}\setminus\{\mathsf{n}\}} \\
\quad \langle \triangleright \mathsf{I}_{\mathit{bag}}(\mathsf{n},\mathtt{this})(\mathsf{bag}) * ((\mathsf{tmp2} = 0 * \exists \mathsf{y}.\ \mathsf{Q}_1(\mathtt{this}, \mathsf{y}) * \mathsf{off.value} \mapsto \mathsf{y}) \vee (\mathsf{tmp2} \neq 0 * \mathsf{P}_1(\mathtt{this})))\rangle^{\mathit{RId}\setminus\{\mathsf{n}\}} \\
\quad \{\mathsf{region}(\{\mathsf{bag}\}, \mathsf{T}_{\mathit{bag}}, \mathsf{n}) * \mathsf{rintr}(\mathsf{I}_{\mathit{bag}}(\mathsf{n},\mathtt{this}), \mathsf{n}) \\
\quad * ((\mathsf{tmp2} = 0 * \exists \mathsf{y}.\ \mathsf{Q}_1(\mathtt{this}, \mathsf{y}) * \mathsf{off.value} \mapsto \mathsf{y}) \vee (\mathsf{tmp2} \neq 0 * \mathsf{P}_1(\mathtt{this})))\} \\
\{\mathsf{bag}(\mathsf{n},\mathtt{this}) * \mathsf{done} = \mathsf{false} \\
\quad * ((\mathsf{tmp2} = 0 * \exists \mathsf{y}.\ \mathsf{Q}_1(\mathtt{this}, \mathsf{y}) * \mathsf{off.value} \mapsto \mathsf{y}) \vee (\mathsf{tmp2} \neq 0 * \mathsf{P}_1(\mathtt{this})))\}
\end{array}
$$

Next, we open the offer region. Since there are four possible offer states, we have to consider four cases. The only interesting case is when the offer is pending, such that `off.state` is 0. To simplify the proof note that there exists an $R : \mathsf{Val} \to \mathsf{Prop}$ such that

$$\mathsf{I}_{\mathit{bag}}(\mathsf{n},\mathtt{this})(\mathsf{bag}) \Leftrightarrow \exists l.\ \mathtt{this}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \mathit{mem}(l) * R(l)$$

In the case where the offer is pending, we thus have:

$$
\begin{array}{l}
\langle \triangleright \mathsf{I}_{\mathit{bag}}(\mathsf{n},\mathtt{this})(\mathsf{bag}) * \mathsf{P}_1(\mathtt{this}) * \mathsf{n} < \mathsf{m} \\
\quad * \ \triangleright \mathsf{I}_{\mathit{offer}}(\mathsf{n}, \mathsf{P}_2, \mathsf{Q}_2, \mathtt{this}, \mathsf{off}, \mathsf{y})(\mathsf{pending}) * \mathsf{off.value} \mapsto \mathsf{y} \rangle \\
\langle \triangleright (\exists l.\ \mathtt{this}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \mathit{mem}(l) * R(l)) * \mathsf{P}_1(\mathtt{this}) * \mathsf{n} < \mathsf{m} \\
\quad * \ \triangleright (\mathsf{off.state} \mapsto 0 * \mathsf{P}_2(\mathtt{this}, \mathsf{y}) * \mathit{spec}(S_2)) * \mathsf{off.value} \mapsto \mathsf{y} \rangle \\
\langle \triangleright (\exists l.\ \mathtt{this}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \mathit{mem}(l) * R(l)) * \mathsf{P}_1(\mathtt{this}) * \mathsf{n} < \mathsf{m} \\
\quad * \ \triangleright (\mathsf{off.state} \mapsto 0) * \triangleright (\mathsf{P}_2(\mathtt{this}, \mathsf{y}) * \mathit{spec}(S_2)) * \mathsf{off.value} \mapsto \mathsf{y} \rangle \\
\quad \langle \triangleright (\mathsf{off.state} \mapsto 0) \rangle \\
\qquad\qquad \mathsf{tmp2} = \mathsf{Interlocked.CompareExchange}\langle \mathbf{int} \rangle(\mathsf{ref\ off.state},\ 1,\ 0); \\
\quad \langle \mathsf{tmp2} = 0 * \mathsf{off.state} \mapsto 1 \rangle^{\mathit{RId}\setminus\{\mathsf{n},\mathsf{m}\}} \\
\langle \mathsf{tmp2} = 0 * \mathsf{off.state} \mapsto 1 * \exists l.\ \mathtt{this}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \mathit{mem}(l) * R(l) * \mathsf{P}_1(\mathtt{this}) * \mathsf{n} < \mathsf{m} \\
\quad * \ \mathsf{P}_2(\mathtt{this}, \mathsf{y}) * \mathsf{spec}(S_2) * \mathsf{off.value} \mapsto \mathsf{y} \rangle^{\mathit{RId}\setminus\{\mathsf{n},\mathsf{m}\}} \\
\langle \triangleright (\exists l.\ \mathtt{this}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \mathit{mem}(l) * R(l)) * \triangleright (\mathsf{off.state} \mapsto 1 * \mathsf{Q}_2(\mathtt{this}, \mathsf{y})) \\
\quad * \ \mathsf{tmp2} = 0 * \exists \mathsf{y}.\ \mathsf{Q}_1(\mathtt{this}, \mathsf{y}) * \mathsf{off.value} \mapsto \mathsf{y} \rangle^{\mathit{RId}\setminus\{\mathsf{n},\mathsf{m}\}}
\end{array}
$$

$$\langle \triangleright \mathsf{I}_{bag}(\mathsf{n}, \mathtt{this})(\mathsf{bag}) * \triangleright \mathsf{I}_{offer}(\mathsf{n}, \mathsf{P}_2, \mathsf{Q}_2, \mathtt{this}, \mathsf{off}, \mathsf{y})(\mathsf{accepted}) \\ * \mathsf{tmp2} = 0 * \exists \mathsf{y}.\ \mathsf{Q}_1(\mathtt{this}, \mathsf{y}) * \mathsf{off.value} \mapsto \mathsf{y} \rangle^{RId \setminus \{\mathsf{n},\mathsf{m}\}}$$

where $S_2 = \forall \mathsf{X}.\ \forall \mathsf{x}, \mathsf{y}.\ \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \mathsf{X} * \mathsf{P}_2(\mathsf{x}, \mathsf{y}) \sqsubseteq^{RId \setminus \mathsf{n}^*} \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} (\mathsf{X} \cup \{\mathsf{y}\}) * \mathsf{Q}_2(\mathsf{x}, \mathsf{y})$.

To finish the above proof we need to show that

$$S_1 \vdash \mathsf{tmp2} = 0 * \mathsf{off.state} \mapsto 1 * \exists l.\ \mathtt{this}_{\mathsf{cont}} \overset{1/2}{\longmapsto} mem(l) * R(l) * \mathsf{P}_1(\mathtt{this}) * \mathsf{n} < \mathsf{m} \\ * \ \mathsf{P}_2(\mathtt{this}, \mathsf{y}) * \mathsf{spec}(S_2) * \mathsf{off.value} \mapsto \mathsf{y} \\ \sqsubseteq^{RId \setminus \{\mathsf{n},\mathsf{m}\}} \triangleright(\exists l.\ \mathtt{this}_{\mathsf{cont}} \overset{1/2}{\longmapsto} mem(l) * R(l)) * \triangleright(\mathsf{off.state} \mapsto 1 * \mathsf{Q}_2(\mathtt{this}, \mathsf{y})) \\ \mathsf{tmp2} = 0 * \exists \mathsf{y}.\ \mathsf{Q}_1(\mathtt{this}, \mathsf{y}) * \mathsf{off.value} \mapsto \mathsf{y}$$

where $S_1 = \forall \mathsf{X}.\ \forall \mathsf{x}, \mathsf{y}.\ \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} (\{\mathsf{y}\} \cup \mathsf{X}) * \mathsf{P}_1(\mathsf{x}) \sqsubseteq^{RId \setminus \mathsf{n}^*} \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \mathsf{X} * \mathsf{Q}_1(\mathsf{x}, \mathsf{y})$. This follows easily using $S_1$ and $S_2$.

# 5 Topos of Trees

In this section we prove some additional results about the internal language of the topos of trees that we use to reason about the iCAP model in Section 1. In particular, we prove that the type of instrumented states and the type of iCAP assertions are total (Section 5.2), and we develop a theory of upwards-closed assertions (Section 5.3).

## 5.1 Löb induction

In this section we develop a simple theory of Löb induction on higher-order arguments.

**Lemma 38.**

$$\forall \phi : \Omega \to \Omega.\ \phi(\bot) \land (\forall s : \Omega.\ \phi(\triangleright s)) \Rightarrow (\forall s : \Omega.\ \phi(s))$$

*Proof.* We prove this externally using the Kripke-Joyal forcing semantics. We thus need to prove that

$$\forall n \in \mathbb{N}.\ n \models \forall \phi : \Omega \to \Omega.\ \phi(\bot) \land (\forall s : \Omega.\ \phi(\triangleright s)) \Rightarrow (\forall s : \Omega.\ \phi(s))$$

Let $n \in \mathbb{N}$, $m \leq n$ and $\phi \in \Omega^{\Omega}(m)$ and assume $m \models \phi(\bot)$ and $m \models \forall s : \Omega.\ \phi(\triangleright s)$. Then we have to show that $m \models \forall s : \Omega.\ \phi(s)$.

Let $k \leq m$ and $s \in \Omega(k)$. Then we have to show that $k \models \phi|_k(s)$.

By definition of $\Omega$ and $\Omega^{\Omega}$, $\phi = (\phi_1, ..., \phi_m)$ where $\phi_i : \Omega(i) \to \Omega(i)$ such that all the $\phi_i$ commutes with the restriction maps, and $s \in \{0, ..., k\}$.

Case $s = 0$: by assumption, $m \models \phi(\bot)$ and thus $\phi_m(\bot_m) = \phi_m(0) = m$. Since all the $\phi_i$ commute with the restriction maps it follows that $\phi_k(s) = k$.

Case $s > 0$: then $s' = s - 1 \in \Omega(k)$ and thus, by assumption, $k \models \phi|_k(\triangleright_k(s'))$ and thus $\phi_k(min(k, (s-1)+1)) = k$, which reduces to $\phi_k(s) = k$. □

**Corollary 2.**

$$\forall \phi : \Omega \to \Omega.\ \phi(\bot) \land (\triangleright(\forall s : \Omega.\ \phi(s)) \Rightarrow (\forall s : \Omega.\ \phi(\triangleright s))) \Rightarrow (\forall s : \Omega.\ \phi(s))$$

*Proof.* We proceed by Löb induction on $\forall s : \Omega.\ \phi(s)$. Assume $\triangleright(\forall s : \Omega.\ \phi(s))$. Then, by assumption, $\forall s : \Omega.\ \phi(\triangleright s)$. Hence, by Lemma 38, $\forall s : \Omega.\ \phi(s)$, as $\phi(\bot)$. □

## 5.2 Totality

In this section we develop a small theory of totality and prove that $\mathcal{M}$ and *Prop* are total objects in $\mathcal{S}$.

We use $total(X)$ as a syntactic shorthand for $\forall x : \blacktriangleright X.\ \exists y : X.\ next^X(y) = x$, where $X$ is a type in the topos of trees.

**Lemma 39.** *Let $X$ be a type. Then,*

$$\triangleright\, total(X) \Rightarrow total(\blacktriangleright X)$$

*Proof.* We prove this externally using the Kripke-Joyal forcing semantics.

Let $n \in \mathbb{N}$ s.t. $n \models \triangleright(\forall x : \blacktriangleright X.\ \exists y : X.\ next^X(y) = x)$. Then we have to show that

$$n \models \forall x : \blacktriangleright\blacktriangleright X.\ \exists y : \blacktriangleright X.\ next^{\blacktriangleright X}(y) = x$$

Let $m \leq n$ and $x \in (\blacktriangleright\blacktriangleright X)(m)$.

- Case $n = 1$: then $x \in (\blacktriangleright\blacktriangleright X)(m) = \{*\}$ and we pick $* \in (\blacktriangleright X)(m) = \{*\}$ for $y$.

- Case $n > 1$: then

$$n - 1 \models \forall x : \blacktriangleright X.\ \exists y : X.\ next^X(y) = x$$

  - Case $m > 2$: then $x \in (\blacktriangleright X)(m-1)$ and there exists a $y \in X(m-1)$ s.t.

$$next^X_{m-1}(x) = y$$

    Thus, we pick $y \in (\blacktriangleright X)(m) = X(m-1)$. Then we have to show that

$$next^{\blacktriangleright X}_m(y) = x$$

    which follows easily, as $next^{\blacktriangleright X}_m = next^X_{m-1}$.

  - Case $m = 2$: then $x \in (\blacktriangleright\blacktriangleright X)(m) = \{*\}$ and there exists a $y \in X(m-1) = (\blacktriangleright X)(m)$. Pick $y$. Then

$$next^{\blacktriangleright X}_2(y) = * = x$$

  - Case $m = 1$: then for every $x \in (\blacktriangleright\blacktriangleright X)(m) = \{*\}$ we can pick $* \in (\blacktriangleright X)(m) = \{*\}$ to be $y$.

□

**Lemma 40.** *Let $X$ be a type. Then*

$$total(X) \Rightarrow total(\Delta(\mathbb{N}) \rightharpoonup_{fin} X)$$

*where*

$$\Delta(\mathbb{N}) \rightharpoonup_{fin} X \stackrel{def}{=} \{f : \Delta(\mathbb{N}) \to (1 + X) \mid \exists i : \Delta(\mathbb{N}).\ \forall j : \Delta(\mathbb{N}).\ i < j \Rightarrow f(j) = inl(*)\}$$

*Proof.* We prove this externally using the Kripke-Joyal forcing semantics.

Let $n \in \mathbb{N}$ s.t. $n \models \forall x : \blacktriangleright X.\ \exists y : X.\ next^X(y) = x$. Then we need to show that

$$n \models \forall x : \blacktriangleright(\Delta(\mathbb{N}) \rightharpoonup_{fin} X).\ \exists y : (\Delta(\mathbb{N}) \rightharpoonup_{fin} X).\ next^{\Delta(\mathbb{N}) \rightharpoonup_{fin} X}(y) = x$$

Let $m \leq n$ and $f \in (\blacktriangleright(\Delta(\mathbb{N}) \rightharpoonup_{fin} X))(m)$.

Case $m = 1$: then $f \in (\blacktriangleright(\Delta(\mathbb{N}) \rightharpoonup_{fin} X))(m) = \{*\}$. Furthermore, $\lambda\_.\ inl(*) \in \Delta(\mathbb{N}) \to (1 + X)$, trivially has a finite domain.

Case $m > 1$: then $f \in (\blacktriangleright(\Delta(\mathbb{N}) \rightharpoonup_{fin} X))(m) = (\Delta(\mathbb{N}) \rightharpoonup_{fin} X)(m-1)$. Hence, $f = (f_1, ..., f_{m-1})$ where $f_i : \mathbb{N} \to (1 + X(i))$, all the $f_i$'s commute with the restriction maps, and

$$m - 1 \models \exists i : \Delta(\mathbb{N}).\ \forall j : \Delta(\mathbb{N}).\ i < j \Rightarrow f(j) = inl(*)$$

Hence, there exists an $i \in \mathbb{N}$ such that

$$m - 1 \models \forall j : \Delta(\mathbb{N}).\ i < j \Rightarrow f(j) = inl(*)$$

Let $Y = \{(i, x) \in \mathbb{N} \times X(m-1) \mid f_{m-1}(i) = inr(x)\}$. Then, for every $(i, x) \in Y$ there exists a $y_x \in X(m)$ such that $next^X_m(y_x) = x$. Define $f_m : \mathbb{N} \to (1 + X(m))$ as follows

$$f_m(i) \stackrel{\text{def}}{=} \begin{cases} inr(y_x) & \text{if } f_{m-1}(i) = inr(x) \\ inl(*) & \text{if } f_{m-1}(i) = inl(*) \end{cases}$$

Then $r^{1+X}_{m-1} \circ f_m = f_{m-1}$ and thus $g = (f_1, ..., f_m) \in (\Delta(\mathbb{N}) \to X)(m)$. To show that $g \in (\Delta(\mathbb{N}) \to X)(m)$ we thus have to show that

$$m \models \exists i : \Delta(\mathbb{N}).\ \forall j : \Delta(\mathbb{N}).\ i < j \Rightarrow g(j) = inl(*)$$

Pick $i$. Let $k \leq m$, $j \in \mathbb{N}$ and $o \leq k$ such that $o \models i < j$. Then we need to show that

$$o \models g(j) = inl(*)$$

Case $o = m$: then, by assumption, $o - 1 \models f(j) = inl(*)$ and thus $g_o(j) = inl(*)$.

Case $o < m$: then $g_o = f_o$ and it follows directly from $o \models f(j) = inl(*)$.

Lastly,

$$next^{\Delta(\mathbb{N}) \rightharpoonup_{fin} X}_m(g) = r^{\Delta(\mathbb{N}) \to X}_{m-1}(g) = (\pi_1(g), ..., \pi_{m-1}(g)) = (f_1, ..., f_{m-1}) = f$$

□

**Lemma 41.** *Let $X$ and $Y$ be types and $\leq^X, \leq^Y$ be relations on $X$ and $Y$, respectively, such that $\leq^Y$ is reflexive. Then*

$$(\forall x : \blacktriangleright Y.\ \exists y : Y.\ next^Y(y) = x \wedge \forall z : Y.\ \triangleright (y \leq^Y z) \Rightarrow (y \leq^Y z \vee \triangleright\bot)) \Rightarrow total(X \to_{mon} Y)$$

*where*

$$X \to_{mon} Y \stackrel{def}{=} \{f : X \to Y \mid \forall x_1, x_2 : X.\ x_1 \leq^X x_2 \Rightarrow f(x_1) \leq^Y f(x_2)\}$$

*Proof.* We prove this externally using the Kripke-Joyal forcing semantics.

Let $n \in \mathbb{N}$ s.t.

$$n \models \forall x : \blacktriangleright Y.\ \exists y : Y.\ next^Y(y) = x \wedge \forall z : Y.\ \triangleright (y \leq^Y z) \Rightarrow (y \leq^Y z \vee \triangleright\bot)$$

Then we have to show that

$$n \models \forall x : \blacktriangleright(X \to_{mon} Y).\ \exists y : (X \to_{mon} Y).\ next^{X \to_{mon} Y}(y) = x$$

Let $m \leq n$ and $f \in (\blacktriangleright(X \to_{mon} Y))(m)$.

- Case $m = 1$: then $f \in (\blacktriangleright(X \to_{mon} Y))(m) = \{*\}$. From the assumed totality of $Y$ it follows that there exists an element $y \in Y(1)$. Thus, define $g_1 : X(1) \to Y(1)$ as $g_1(x) = y$. Then $g = (g_1)$ is trivially an element of $(X \to Y)(1)$. To show that $g \in (X \to_{mon} Y)(1)$ we thus have to show that

  $$m \models \forall x_1, x_2 : X.\ x_1 \leq^X x_2 \Rightarrow g(x_1) \leq^Y g(x_2)$$

  Let $x_1, x_2 \in X(1)$ such that $1 \models x_1 \leq^X x_2$. Then we need to show that

  $$1 \models g(x_1) \leq^Y g(x_2)$$

  which reduces to $1 \models y \leq^Y y$ which holds by reflexivity of $\leq^Y$.

- Case $m > 1$: then $f \in (\blacktriangleright(X \to_{mon} Y))(m) = (X \to_{mon} Y)(m-1)$. Hence $f = (f_1, ..., f_{m-1})$ where $f_i : X(i) \to Y(i)$, all the $f_i$'s commute with the restriction maps, and

  $$m - 1 \models \forall x_1, x_2 : X.\ x_1 \leq^X x_2 \Rightarrow f(x_1) \leq^Y f(x_2)$$

  By assumption, for every $x \in X(m-1)$ there exists a $y_x \in Y(m)$ such that

  $$next^Y_m(y_x) = f_{m-1}(x)$$

  and (since $m \not\models \triangleright\bot$)

  $$\forall z \in Y(m).\ (m - 1 \models y_x|_{m-1} \leq^Y z|_{m-1}) \Rightarrow m \models y_x \leq^Y z \qquad (1)$$

Define $f_m : X(m) \to Y(m)$ as follows

$$f_m(x) \stackrel{\text{def}}{=} y_{next^X_m(x)}$$

For all $x \in X(m)$ we have that

$$r^Y_{m-1}(f_m(x)) = r^Y_{m-1}(y_{next^X_m(x)}) = next^Y_m(y_{next^X_m(x)}) = f_{m-1}(next^X_m(x))$$

Thus $g = (f_1, ..., f_m) \in (X \to Y)(m)$. It thus remains to show that

$$m \models \forall x_1, x_2 : X.\ x_1 \leq_X x_2 \Rightarrow g(x_1) \leq_Y g(x_2)$$

Since $g|_{m-1} = f$ it follows easily for step-indexes $k \leq m-1$. Thus, assume $x_1, x_2 \in X(m)$ such that $m \models x_1 \leq^X x_2$. Then we need to prove that $m \models g(x_1) \leq^Y g(x_2)$. Unfolding, this reduces to

$$m \models y_{next^X_m(x_1)} \leq^Y y_{next^X(m)(x_2)}$$

Since $y_{next^X_m(x_i)}|_{m-1} = next^Y_m(y_{x_i|_{m-1}}) = f_{m-1}(x_i|_{m-1})$, by (1), this reduces to

$$m-1 \models f(x_1|_{m-1}) \leq_Y f(x_2|_{m-1})$$

which holds by downwards-closure of $m \models x_1 \leq^X x_2$.

□

**Lemma 42.** *Let $X$ be an object in Sets and $\leq : \Delta(X) \times \Delta(X) \to \Omega \in \mathcal{S}$. Then*

$$\forall x : \blacktriangleright \mathcal{P}^\uparrow(\Delta(X)).\ \exists y : \mathcal{P}^\uparrow(\Delta(X)).$$
$$next^{\mathcal{P}^\uparrow(\Delta(X))}(y) = x \wedge \forall z : \mathcal{P}^\uparrow(\Delta(X)).\ \triangleright (y \leq^{\mathcal{P}^\uparrow(\Delta(X))} z) \Rightarrow (y \leq^{\mathcal{P}^\uparrow(\Delta(X))} z \vee \triangleright \bot)$$

*where*

$$\mathcal{P}^\uparrow(X) \stackrel{def}{=} \{p : X \to \Omega \mid \forall x, y : X.\ x \leq y \wedge p(x) \Rightarrow p(y)\}$$

*and*

$$p \leq^{\mathcal{P}^\uparrow(\Delta(X))} q \quad \textit{iff} \quad \forall x : \Delta(X).\ p(x) \Rightarrow q(x)$$

*Proof.* We prove this externally using the Kripke-Joyal forcing semantics.

Let $n \in \mathbb{N}$ and $p \in (\blacktriangleright \mathcal{P}^\uparrow(\Delta(X)))(n)$. Then we need to show that

$$n \models \exists y : \mathcal{P}^\uparrow(\Delta(X)).\ next^{\mathcal{P}^\uparrow(\Delta(X))}(y) = x \wedge \forall z : \mathcal{P}^\uparrow(\Delta(X)).\ \triangleright (y \leq z) \Rightarrow (y \leq z \vee \triangleright \bot)$$

- Case $n = 1$: then $p = *$. To construct an element $q \in (\mathcal{P}^{\uparrow}(\Delta(X)))(1)$ take $q : X \to \Omega(1)$ to be $q(x) = 0$. Then

$$n \models \forall x, y : X.\ x \leq y \wedge q(x) \Rightarrow q(y)$$

holds trivially as $q(x)$ is false for all $x$ at step-index 1.

It thus remains to show that

$$n \models next^{\mathcal{P}^{\uparrow}(\Delta(X))}(q) = p \wedge \forall r : \mathcal{P}^{\uparrow}(\Delta(X)).\ \triangleright (q \leq^{\mathcal{P}^{\uparrow}(\Delta(X))} r) \Rightarrow (q \leq^{\mathcal{P}^{\uparrow}(\Delta(X))} r \vee \triangleright \bot)$$

The first conjunction holds trivially, as $next_1^{\mathcal{P}^{\uparrow}(\Delta(X))}(q) = * = p$. To show the second conjunction let $r \in \mathcal{P}^{\uparrow}(\Delta(X))$ s.t. $n \models \triangleright(q \leq^{\mathcal{P}^{\uparrow}(\Delta(X))} r)$. It thus remains to show that

$$n \models (\forall x : \Delta(X).\ q(x) \Rightarrow r(x)) \vee \triangleright \bot$$

which again holds trivially, as $q(x)$ is false for all $x$ at step-index 1.

- Case $n > 1$: then $p \in (\mathcal{P}^{\uparrow}(\Delta(X)))(n-1)$. Thus $p = (p_1, ..., p_{n-1})$ where $p_i : X \to \Omega(i)$ such that all the $p_i$'s commute with the restriction maps and

$$n - 1 \models \forall x, y : X.\ x \leq y \wedge p(x) \Rightarrow p(y)$$

Define $p_n : X \to \Omega(n)$ as $p_n(x) = p_{n-1}(x)$. Then $q = (p_1, ..., p_n) \in (\Delta(X) \to \Omega)(n)$. To show that $q \in \mathcal{P}^{\uparrow}(\Delta(X))$ it thus remains to show that

$$n \models \forall x, y : X.\ x \leq y \wedge q(x) \Rightarrow q(y)$$

Let $m \leq n$ and $x, y \in X(m)$ such that $m \models x \leq y$ and $m \models q|_m(x)$. Since $m \models q|_m(x)$ it follows that $m \neq n$ as otherwise $p_n(x) = p_{n-1}(x) = n$, which is a contradiction. Hence, $m \models q|_m(y)$ as $q|_m = p|_m$.

Lastly, we need to show that

$$n \models next^{\mathcal{P}^{\uparrow}(\Delta(X))}(q) = p \wedge \forall r : \mathcal{P}^{\uparrow}(\Delta(X)).\ \triangleright (q \leq^{\mathcal{P}^{\uparrow}(\Delta(X))} r) \Rightarrow (q \leq^{\mathcal{P}^{\uparrow}(\Delta(X))} r \vee \triangleright \bot)$$

The first conjunction holds as

$$next_m^{\mathcal{P}^{\uparrow}(\Delta(X))}(q) = q|_{m-1} = p|_{m-1}$$

for every $m \leq n$. For the second conjunct, let $r \in (\mathcal{P}^{\uparrow}(\Delta(X)))(n)$ and $m \leq n$ s.t.

$$m \models \triangleright(q \leq^{\mathcal{P}^{\uparrow}(\Delta(X))} r)$$

– Case $m = 1$: then
$$m \models q \leq^{\mathcal{P}^\uparrow(\Delta(X))} r \vee \triangleright\bot$$
holds easily, by the right-hand side of the disjunction.
– Case $m > 1$: then by assumption $m - 1 \models q|_{m-1} \leq^{\mathcal{P}^\uparrow(\Delta(X))} r|_{m-1}$. To show $m \models q \leq^{\mathcal{P}^\uparrow(\Delta(X))} r$ assume $x \in X$ and $k \leq m$ such that $k \models q|_k(x)$.
  * Case $k = m$: then $q|_m(x) = p_{m-1}(x) = m$, which is a contradiction as $p_{m-1}(x) \in \Omega(m-1)$.
  * Case $k < m$: then $k \models r|_k(x)$ follows from the $m - 1 \models q|_{m-1} \leq r|_{m-1}$ assumption

□

**Lemma 43.** *Let $X$ be an object in $\mathcal{S}$ and $\leq : X \times X \to \Omega$. Then total$(\mathcal{P}^\uparrow(X))$ where*
$$\mathcal{P}^\uparrow(X) = \{p : X \to \Omega \mid \forall x, y : X.\ x \leq y \wedge p(x) \Rightarrow p(y)\}$$

*Proof.* We prove this externally using the Kripke-Joyal forcing semantics.

Let $n \in \mathsf{N}$ and $p \in (\mathcal{P}^\uparrow(X))(n)$.

- Case $n = 1$: then $p = *$; take $q_1 : X(1) \to \Omega(1)$ to be $q_1(x) = 0$ then $q = (q_1) \in (X \to \Omega)(1)$. Thus, it remains to show that
$$1 \models \forall x, y : X.\ x \leq y \wedge q(x) \Rightarrow q(y)$$
Thus, let $x, y \in X(1)$ such that $1 \models x \leq y$ and $1 \models q(x)$. Then $q_1(x) = 1$, which is a contradiction.

- Case $n > 1$: then $p \in (\mathcal{P}^\uparrow(X))(n-1)$. Hence $p = (p_1, ..., p_{n-1})$ where $p_i : X(i) \to \Omega(i)$ such that
$$\forall i \in \{1, ..., n-2\}.\ p_i \circ r_i^X = r_i^\Omega \circ p_{i+1}$$
and
$$n - 1 \models \forall x, y : X.\ x \leq y \wedge p(x) \Rightarrow p(y)$$
Define $p_n : X(n) \to \Omega(n)$ as $p_n(x) = p_{n-1}(r_{n-1}^X(x))$. To show that $p' = (p_1, ..., p_n) \in (X \to \Omega)(n)$ we thus have to show that
$$\forall i \in \{1, ..., n-1\}.\ p'_i \circ r_i^X = r_i^\Omega \circ p'_{i+1}$$
This follows easily from the above assumption for $i \in \{1, ..., n-2\}$ as $p'_i = p_i$ and $p'_{i+1} = p_{i+1}$. It thus remains to show that $p'_{n-1} \circ r_{n-1}^X = r_{n-1}^\Omega \circ p'_n$. Let $x \in X(n)$. Then
$$r_{n-1}^\Omega(p'_n(x)) = r_{n-1}^\Omega(p_{n-1}(r_{n-1}^X(x))$$
Since $p_{n-1} : X(n-1) \to \Omega(n-1)$ it follows that $r_{n-1}^\Omega \circ p_{n-1} = p_{n-1}$.

It thus remains to prove that

$$n \models \forall x, y : X.\ x \leq y \wedge p'(x) \Rightarrow p'(y)$$

Let $m \leq n$, $x \in X(m)$, $k \leq m$, $y \in X(k)$, $j \leq k$ such that

$$j \models x|_j \leq_j y|_j \qquad\qquad j \models p'|_j(x|_j)$$

- Case $j = n$: then $p'_n(x) = p_{n-1}(r^X_{n-1}(x)) = n$ which is a contradiction.
- Case $j < n$: then $p'|_j = p|_j$ and thus $j \models p|_j(x|_j)$ from which it follows that $j \models p|_j(y|_j)$ and thus $j \models p'|_j(y|_j)$.

□

**Lemma 44.** *Let $X$ be a total and inhabited object in $\mathcal{S}$. Then*

$$\forall p \in \Omega.\ \forall q \in X \to \Omega.\ (p \Rightarrow \exists x : X.\ q(x)) \Rightarrow (\exists x : X.\ p \Rightarrow q(x))$$

*Proof.* We prove this externally using the Kripke-Joyal forcing semantics. We thus have to prove that

$$\forall n \in \mathbb{N}.\ n \models \forall p \in \Omega.\ \forall q \in X \to \Omega.\ (p \Rightarrow \exists x : X.\ q(x)) \Rightarrow (\exists x : X.\ p \Rightarrow q(x))$$

Unfolding, this reduces to

$$\begin{aligned}\forall n \in \mathbb{N}.\ \forall m \leq n.\ \forall p \in \Omega(m).\ \forall i \leq m.\ \forall q \in \Omega^X(i).\ \forall j \leq i.\\ (j \models p|_j \Rightarrow \exists x : X.\ q|_j(x)) \Rightarrow (j \models \exists x : X.\ p|_j \Rightarrow q|_j(x))\end{aligned}$$

Case $\forall k \leq j.\ k \not\models p|_j$: Since $X$ is inhabited there exists a global object $y : 1 \Rightarrow X$. Pick $x \in X(j)$ to be $y_j(*)$.

Case $\exists k \leq j.\ k \models p|_j$: Let $k = \max\{k \in \mathsf{N} \mid k \leq j \wedge k \models p|_j\}$. Then, by definition, $k \models p|_j$ and thus $k \models \exists x : X.\ q|_j(x)$. Hence, there exists an $x \in X(k)$ such that $k \models q|_k(x)$. Since $X$ is total, there thus exists a $y \in X(j)$ such that $y|_k = x$. □

**Lemma 45.**

$$\mathit{total}(\mathcal{M}) \wedge \mathit{total}(\mathit{Prop})$$

*Proof.* Totality of $\mathcal{M}$ reduces to totality of *RIntr*, which reduces to totality of

$$\blacktriangleright((\Delta(\mathit{SId}) \times (\Delta(\mathit{RId}) \rightharpoonup_{fin} \mathit{RIntr})) \to_{mon} \mathcal{P}^{\uparrow}(\Delta(\mathit{AState})))$$

By Lemma 39, this reduces to totality of

$$(\Delta(\mathit{SId}) \times (\Delta(\mathit{RId}) \rightharpoonup_{fin} \mathit{RIntr})) \to_{mon} \mathcal{P}^{\uparrow}(\Delta(\mathit{AState}))$$

later. This follows easily by Lemmas 41 and 42.

Totality of *Prop* follows from Lemma 43. □

## 5.3 Upwards-closed assertions

In this section we develop a small theory of upwards-closed assertions and relate them to $\neg\neg$-closed assertions. Upwards-closed assertions are assertions that are upwards-closed in the step-index (Lemma 46). Upwards-closed assertions are thus false at every step index or true at every step index (Lemma 47). This theory is used to prove that the composition operator on local states is upwards-closed, which in turn is used to prove that $\triangleright$ commutes over $*$ (See Lemma 12).

*Upwards-closure* $\boxed{up : \Omega \to \Omega \in \mathcal{S}}$

$$up(p : \Omega) \stackrel{\text{def}}{=} \triangleright p \Rightarrow (p \vee \triangleright \bot)$$

**Lemma 46.**

$$\forall n \in \mathbb{N}.\ \forall p \in \Omega(n).\ (n \models up(p)) \Leftrightarrow (\forall m < n.\ m \models p \Rightarrow m + 1 \models p)$$

*Proof.*

$$\begin{aligned}
(n \models up(p)) &\Leftrightarrow (\forall m \leq n.\ (m \models \triangleright p) \Rightarrow (m \models p \vee \triangleright \bot)) \\
&\Leftrightarrow ((1 \models \triangleright p) \Rightarrow (1 \models p \vee \triangleright \bot)) \wedge \\
&\qquad ((\forall m < n.\ (m + 1 \models \triangleright p) \Rightarrow (m + 1 \models p \vee \triangleright \bot))) \\
&\Leftrightarrow (1 \models p \vee \triangleright \bot) \wedge (\forall m < n.\ (m \models p) \Rightarrow (m + 1 \models p \vee \triangleright \bot)) \\
&\Leftrightarrow ((1 \models p) \vee \top) \wedge (\forall m < n.\ (m \models p) \Rightarrow ((m + 1 \models p) \vee \bot)) \\
&\Leftrightarrow \forall m < n.\ (m \models p) \Rightarrow (m + 1 \models p)
\end{aligned}$$

□

**Lemma 47.**

$$\forall n \in \mathbb{N}.\ \forall p \in \Omega(n).\ (n \models up(p)) \Leftrightarrow ((\forall m \in \mathbb{N}.\ m \not\models p) \vee n \models p)$$

**Lemma 48.**

$$\forall n \in \mathbb{N}.\ \forall p \in \Omega(n).\ (n \models up(p)) \Rightarrow (n \models p \vee \neg p)$$

*Proof.* Assume $n \models up(p)$. The by Lemma 46 it follows that

$$\forall m < n.\ (m \models p) \Rightarrow (m + 1 \models p) \tag{2}$$

Case $n \models p$: Then $n \models p \vee \neg p$ trivially.
Case $n \not\models p$: Assume $k \leq n$ such that $k \models p$ then by (2) it follows that $n \models p$, which is a contradiction. Hence, $n \models \neg p$. □

**Lemma 49.**

$$\forall p \in \Omega.\ up(p) \Rightarrow p \vee \neg p$$

**Lemma 50.**

$$\forall n \in \mathbb{N}.\ \forall p \in \Omega.\ (n \models \neg\neg p) \Leftrightarrow (\exists k \leq n.\ k \models p)$$

*Proof.*

$$\begin{aligned}
(n \models \neg\neg p) &\Leftrightarrow (\forall m \leq n.\ (m \models p \Rightarrow \bot) \Rightarrow (m \models \bot)) \\
&\Leftrightarrow (\forall m \leq n.\ \neg(\forall k \leq m.\ (k \models p) \Rightarrow k \models \bot)) \\
&\Leftrightarrow (\forall m \leq n.\ \neg(\forall k \leq m.\ \neg(k \models p))) \\
&\Leftrightarrow (\forall m \leq n.\ \exists k \leq m.\ \neg\neg(k \models p)) \\
&\Leftrightarrow (\forall m \leq n.\ \exists k \leq m.\ k \models p) \\
&\Leftrightarrow \exists k \leq n.\ k \models p
\end{aligned}$$

□

**Lemma 51.**

$$\forall p \in \Omega.\ up(p) \Rightarrow (p \Leftrightarrow \neg\neg p)$$

*Proof.* Proved externally using the Kripke-Joyal forcing semantics.

Let $n \in \mathbb{N}$, $p \in \Omega(n)$ such that $n \models up(p)$. Then

$$\forall m < n.\ m \models p \Rightarrow m + 1 \models p$$

and we have to show that

$$n \models (p \Rightarrow \neg\neg p) \qquad\qquad n \models (\neg\neg p \Rightarrow p)$$

By Lemma 50 this reduces to

$$\forall m \leq n.\ (m \models p) \Rightarrow (\exists k \leq m.\ k \models p)$$

and

$$\forall m \leq n.\ (\exists k \leq m.\ k \models p) \Rightarrow (m \models p)$$

The first proof obligation follows trivially, by picking $k$ to be $m$. The second proof obligation follows by repeat application of the upwards-closure assumption. □

**Lemma 52.**

$$\forall x, y : \Delta(X).\ up(x =_{\Delta(X)} y)$$

*Proof.* Proved externally using the Kripke-Joyal forcing semantics. □

**Lemma 53.**

$$\forall x, y : \Delta(X).\ x =_{\Delta(X)} y \vee \neg(x =_{\Delta(X)} y)$$

*Proof.* Proved externally using the Kripke-Joyal forcing semantics. □

**Lemma 54.**

$$up(\bot) \wedge up(\top)$$

*Proof.* $up(\bot)$ and $up(\top)$ holds trivially, as

$$up(\bot) = (\triangleright\bot \Rightarrow \bot \vee \triangleright\bot) \qquad up(\top) = (\triangleright\top \Rightarrow \top \vee \triangleright\bot)$$

□

**Lemma 55.**

$$\forall p, q : \Omega.\ up(p) \wedge up(q) \Rightarrow (up(p \wedge q) \wedge up(p \vee q))$$

*Proof.* Assume $\triangleright(p \wedge q)$. Then $\triangleright p \wedge \triangleright q$ and thus $(p \vee \triangleright\bot) \wedge (q \vee \triangleright\bot)$, which implies that $(p \wedge q) \vee \triangleright\bot$.

Assume $\triangleright(p \vee q)$. Then $\triangleright p \vee \triangleright q$ and thus $(p \vee \triangleright\bot) \vee (q \vee \triangleright\bot)$, which implies that $(p \vee q) \vee \triangleright\bot$. □

**Lemma 56.**

$$\forall p, q \in \Omega.\ up(q) \Rightarrow up(p \Rightarrow q)$$

*Proof.* Proved externally using the Kripke-Joyal forcing semantics.

Let $n \in \mathbb{N}$, $p, q \in \Omega(n)$ such that $n \models up(q)$. Then

$$\forall m < n.\ (m \models q) \Rightarrow (m + 1 \models q)$$

and we have to show that

$$\forall m < n.\ (m \models p \Rightarrow q) \Rightarrow (m + 1 \models p \Rightarrow q)$$

Assume $m < n$ such that $m \models p \Rightarrow q$. Then

$$\forall k \leq m.\ k \models p \Rightarrow k \models q$$

Let $j \leq m+1$ such that $j \models p$. If $j \leq m$ then $j \models q$ trivially. Otherwise, if $j = m+1$, then by downwards-closure, $m \models p$. Thus, $m \models q$ and by upwards-closure of $q$, $m+1 \models q$. □

**Lemma 57.** *Let $\tau$ be a total and inhabited type.*

$$\forall p : \tau \to \Omega.\ (\forall x : \tau.\ up(p(x))) \Rightarrow up(\exists x : \tau.\ p(x))$$

*Proof.* Assume $\triangleright \exists x : \tau.\ p(x)$. Since $\tau$ is total and inhabited, $\exists x : \tau.\ \triangleright p(x)$. Thus, $\exists x : \tau.\ (p(x) \vee \triangleright \bot)$, from which it follows that $(\exists x : \tau.\ p(x)) \vee \triangleright \bot$. □

**Lemma 58.** *Let $\tau$ be an arbitrary type.*

$$\forall p : \tau \to \Omega.\ (\forall x : \tau.\ (p(x) \vee \neg p(x))) \Rightarrow (\forall x : \tau.\ p(x)) \vee (\exists x : \tau.\ \neg p(x))$$

*Proof.* We prove this externally, using the Kripke-Joyal forcing semantics.

Assume $n \in \mathbb{N}$, $p \in [\![\Omega^\tau]\!](n)$, and $m \leq n$ such that

$$m \models \forall x : \tau.\ (p|_m(x) \vee \neg p|_m(x))$$

then we need to show that

$$m \models (\forall x : \tau.\ p|_m(x)) \vee (\exists x : \tau.\ p|_m(x))$$

From the assumption, it follows that

$$\forall x \in [\![\tau]\!](m).\ (m \models p|_m(x)) \vee (\forall k \leq m.\ k \not\models p|_k(x|_k))$$

Case $\forall x \in [\![\tau]\!](m).\ m \models p|_m(x)$: then $m \models \forall x : \tau.\ p|_m(x)$. Case $\exists x \in [\![\tau]\!](m).\ m \not\models p|_m(x)$: hence, by assumption,

$$\forall k \leq m.\ k \not\models p|_k(x|_k)$$

Hence, $m \models \exists : \tau.\ p|_m(x)$. □

**Lemma 59.** *Let $\tau$ be an arbitrary type.*

$$\forall p : \tau \to \Omega.\ (\forall x : \tau.\ up(p(x))) \Rightarrow up(\forall x : \tau.\ p(x))$$

*Proof.* Assume $\triangleright \forall x : \tau.\ p(x)$. Then $\forall x : \tau.\ \triangleright p(x)$ and thus

$$\forall x : \tau.\ (p(x) \vee \triangleright \bot)$$

Since by assumption, $\forall x : \tau.\ up(p(x))$ it follows by Lemma 49 that

$$\forall x : \tau.\ p(x) \vee \neg p(x)$$

Hence, by Lemma 58,

$$(\forall x : \tau.\ p(x)) \vee (\exists x : \tau.\ \neg p(x))$$

If $\forall x : \tau.\ p(x)$ then $(\forall x : \tau.\ p(x)) \vee \triangleright \bot$ holds trivially. If $\exists x : \tau.\ \neg p(x)$ then $\triangleright \bot$. □

**Lemma 60.**

$$\{f : \Delta(X) \to \Omega \mid \forall x : \Delta(X).\ up(f(x))\} \cong \Delta(X \to \Omega)$$

*Proof.* Define $\phi : (\{f : \Delta(X) \to \Omega \mid \forall x : \Delta(X).\ up(f(x))\}) \cong \Delta(X \to \Omega) : \psi$ as follows

$$\phi_n((f_1, ..., f_n)) = f_1$$
$$\psi_n(f) = (f^1, ..., f^n)$$

where $f^i \in (\Delta(X) \to \Omega)(i)$ is defined as follows

$$f^i(x) = \begin{cases} i & \text{if } f(x) = 1 \\ 0 & \text{otherwise} \end{cases}$$

To show that $\psi$ is well-defined we have to show that

$$\forall n \in \mathbb{N}.\ \psi_n \circ r^n_{\Delta(X \to \Omega)} = r^n_{\Delta(X) \to \Omega} \circ \psi_{n+1}$$

and

$$\forall f \in X \to \Omega.\ \forall n \in \mathbb{N}.\ n \models \forall x : \Delta(X).\ up(\psi_n(f)(x))$$

The first obligation follows easily, as

$$r^n_{\Delta(X) \to \Omega}(\psi_{n+1}(f)) = r^n_{\Delta(X) \to \Omega}((f^1, ..., f^{n+1})) = (f^1, ..., f^n) = \psi_n(f) = \psi_n(r^n_{\Delta(X \to \Omega)}(f))$$

for $f \in X \to \Omega$.

For the second obligation, let $m \leq n$ and $x \in X$. Then we need to show that $m \models up(\psi_n(f)|_m(x))$. Let $k < m$ such that $k \models \psi_n(f)|_k(x|_k)$, by Lemma 46 it thus suffices to show that $k + 1 \models \psi_n(f)|_{k+1}(x|_{k+1})$. Since $k \models \psi_n(f)|_k(x|_k)$ it follows that $f^k(x) = k$ and thus that $f(x) = 1$ and $f^{k+1}(x) = k+1$. Hence, $k+1 \models \psi_n(f)|_{k+1}(x|_{k+1})$.

Clearly, $\psi_n \circ \phi_n = id_n$ for all $n \in \mathbb{N}$. It thus only remains to show that $\phi_n \circ \psi_n = id_n$. Let $n \in \mathbb{N}$ and $f \in (\{f : \Delta(X) \to \Omega \mid \forall x : \Delta(X).\ up(f(x))\})(n)$. Then $f = (f_1, ..., f_n)$ and

$$n \models \forall x : \Delta(X).\ up(f(x))$$

Hence, by the forcing semantics and Lemma 46,

$$\forall x \in X.\ \forall m < n.\ m \models f|_m(x|_m) \Rightarrow m + 1 \models f|_{m+1}(x|_{m+1})$$

Let $x \in X$. If $f_1(x) = 0$ then

$$f_2(x) = f_3(x) = ... = f_n(x) = 0 = f^1_1(x) = f^2_1(x) = ... = f^n_1(x)$$

Conversely, if $f_1(x) = 1$ then

$$f_2(x) = 2 = f^2_1(x), \quad f_3(x) = 3 = f^3_1(x), ..., f_n(x) = n = f^n_1(x)$$

Thus,

$$\phi_n(\psi_n(f)) = \phi_n(f_1) = (f^1_1, f^2_1, ..., f^n_1) = (f_1, ..., f_n) = f$$

□

# 6 Mini C♯

In this section we define the syntax and semantics of a subset of C♯, dubbed mini C♯. In addition to the basic object-oriented and imperative features of C♯, this subset includes named delegates, fork concurrency, and compare-and-swap. We use a restricted syntax to simplify the presentation of the proof system.

## 6.1 Syntax

The syntax of the language is given below. In the syntax we use the following metavariables: f ranges over field names, m over method names, C over class names, x, y, z, o, and n over program variables. We denote the set of field names by *FName*, the set of method names by *MName*, the set of class names by *CName*, and the set of statements by *Stm*. We use an overbar for sequences.

| | | | |
|---|---|---|---|
| L | ::= | `class` C $\{\overline{\mathsf{Cf}}; \bar{\mathsf{M}}\}$ | Class definition |
| M | ::= | C m$(\overline{\mathsf{Cx}})$ $\{\overline{\mathsf{Cy}}; \bar{\mathsf{s}};$ `return` z$\}$ | Method definition |
| s | ::= | | Statement |
| | | x = y | assignment |
| | | x = `null` | initialization |
| | | x = y.f | field access |
| | | x.f = y | field update |
| | | `if` (x == y) $\{\bar{\mathsf{s}}_1\}$ `else` $\{\bar{\mathsf{s}}_2\}$ | conditional |
| | | x = `new` C() | object creation |
| | | x = `delegate` y.m | named delegate |
| | | x = y.m$(\bar{\mathsf{z}})$ | method invocation |
| | | x = y$(\bar{\mathsf{z}})$ | delegate application |
| | | `fork`(x) | fork process |
| | | x = `CAS`(y.f, o, n) | compare-and-swap |

A new thread is forked by calling `fork` with a reference to a named delegate referring to a method with no parameters. The language does not feature a join statement[1], and the return value of a forked delegate is simply ignored. Each thread has a private stack, but all threads share a common heap.

The compare-and-swap statement, x = `CAS`(y.f, o, n), atomically compares the value of field f of object y with the value of o, and updates it with the value of n, if they match. In this case `CAS` also sets x to y; otherwise, `CAS` sets x to `null`.

The statement syntax does not include sequential composition. Instead we take the body of a method to be a sequence of statements. We use ; for concatenation of statement sequences. Hence, when we write $s_1; s_2$ for $s_1, s_2 \in \mathit{seq}\ \mathit{Stm}$, it does not implicitly follow that $s_1$ is non-empty.

[1]However, using compare-and-swap it is possible to code join-like functionality, and the logic is sufficiently strong to reason about such an encoding.

## 6.2 Operational Semantics

The operational semantics is a small-step interleaving semantics, giving a strong memory model. The semantic domains used in the definition of the operational semantics are defined below. We assume disjoint countably infinite sets of thread identifiers ($TId$), object identifiers ($OId$), and closure identifiers ($CId$).

$$
\begin{array}{rll}
t \in TId & & \text{thread identifiers} \\
o \in OId & & \text{object identifiers} \\
c \in CId & & \text{closure identifiers} \\
v \in CVal ::= null \mid o \mid c & & \text{C}^\sharp \text{ values} \\
OHeap \stackrel{\text{def}}{=} OId \times FName \rightharpoonup_{fin} CVal & & \text{object heap} \\
THeap \stackrel{\text{def}}{=} OId \rightharpoonup_{fin} CName & & \text{type heap} \\
CHeap \stackrel{\text{def}}{=} CId \rightharpoonup_{fin} OId \times MName & & \text{closure heap} \\
h \in Heap \stackrel{\text{def}}{=} OHeap \times THeap \times CHeap & & \text{heap} \\
l \in Stack \stackrel{\text{def}}{=} Var \rightharpoonup_{fin} CVal & & \text{stack} \\
TCStack \stackrel{\text{def}}{=} seq\ (Stm + (Stack \times Var \times Var)) & & \text{thread call stack} \\
x, y, z \in Thread \stackrel{\text{def}}{=} TId \times Stack \times TCStack & & \text{thread} \\
T \in TPool \stackrel{\text{def}}{=} \{U \in \mathcal{P}_{fin}(Thread) \mid utid(U)\} & & \text{thread pool} \\
Prg \stackrel{\text{def}}{=} (Heap \uplus \{\lightning\}) \times TPool & & \text{program}
\end{array}
$$

Here $utid$ expresses that thread identifiers are unique in a given thread pool:

$$utid(U) = \forall x, y \in U.\ x.t = y.t \Rightarrow x = y \qquad \text{for } U \in \mathcal{P}_{fin}(Thread)$$

We use $x.t$, $x.l$, and $x.s$ to refer to the first, second and third component of a thread $x \in Thread$. We use $h.o$, $h.t$ and $h.c$ to refer to the object, type and closure heap of a heap $h \in Heap$.

A normal machine configuration consists of a heap and a pool of threads. The heap is global and shared between every thread. Each thread consists of a thread identifier, a private stack, and a call stack. Since we are using a small-step semantics, the call stack includes explicit returns to restore the stack at the end of method calls. The call stack is thus a sequence of programming language statements and method returns. Method returns consists of the stack prior to the method call, the variable the method return value should be assigned to, and the variable containing the method return value. We use $return\ (l, x, y)$ as notation for method returns. Formally, $return\ (l, x, y)$ is notation for $inr(l, x, y) \in Stm + (Stack \times Var \times Var)$. We use $stm(s)$ as notation for $map(inl)(s) \in TCStack$ when $s \in Stm$. We use $\cdot$ for cons-ing on call stacks and ; for concatenation of call stacks.

A faulty machine configuration consists of a special fault heap, $\lightning$, and a thread pool. We use this faulty configuration to indicate memory errors, such as an attempt to access a field that does not exist.

The presentation of the operational semantics is inspired by the Views framework [2], and factors the small-step program evaluation relation, $\rightarrow$, into a labelled thread pool evaluation relation, $\xrightarrow{a}$, and an action semantics, $[\![-]\!]$. The labelled thread pool evaluation relation, $\xrightarrow{a}$, defines the local effects (i.e., stack effects) of executing a single thread for one step of execution. The action semantics defines the global effects (i.e., heap effects) of executing an atomic action. These are related through the action label ($a$) of the thread pool evaluation relation. This factorization simplifies the definition of safety in Section 1. Due to this factorization, the labelled thread pool evaluation relation non-deterministically "guesses" the right values when reading from and allocating on the heap. For instance, the READ rule "guesses" the current value $v$ of the given field in the heap. The semantics of the $read(-)$ action then enforces that the "guess" was correct, through the STEP rule.

*Actions* $\boxed{Act \in \text{Sets}}$

$$\begin{array}{rcl} a \in Act & ::= & id \mid read(o, f, v) \mid write(o, f, v) \mid cas(o, f, v_o, v_n, r) \\ & \mid & oalloc(T, o) \mid calloc(o, m, c) \mid otype(o, T) \mid ctype(c, T, o, m) \mid \lightning \end{array}$$

where $c \in CId$, $o \in OId$, $f \in FName$, $v, v_o, v_n, r \in CVal$, $T \in CName$, $m \in MName$.

*Single-step semantics of non-forking statements*

$$\boxed{\rightarrow \subseteq (Stack \times (Stm + (Stack \times Var \times Var))) \times Act \times (Stack \times TCStack)}$$

We use $\mathsf{body}(T, \mathsf{m})$ to refer to the body of method $\mathsf{m}$ from the $T$ class. We use $\mathsf{fields}(T)$ to refer to the names of the fields defined by class $T$. Naturally, both $\mathsf{body}$ and $\mathsf{fields}$ are partial functions.

$$\frac{\mathsf{x}, \mathsf{y} \in dom(l)}{(l, inl(\mathsf{x} = \mathsf{y})) \xrightarrow{id} (l[\mathsf{x} \mapsto l(\mathsf{y})], \varepsilon)} \textsc{Assign}$$

$$\frac{\mathsf{x}, \mathsf{y} \in dom(l) \qquad l(\mathsf{x}) \in OId \qquad v \in CVal}{(l, inl(\mathsf{y} = \mathsf{x.f})) \xrightarrow{read(l(\mathsf{x}), \mathsf{f}, v)} (l[\mathsf{y} \mapsto v], \varepsilon)} \textsc{Read}$$

$$\frac{\mathsf{x}, \mathsf{y} \in dom(l) \qquad l(\mathsf{x}) \in OId}{(l, inl(\mathsf{x.f} = \mathsf{y})) \xrightarrow{write(l(\mathsf{x}), \mathsf{f}, l(\mathsf{y}))} (l, \varepsilon)} \textsc{Write}$$

$$\frac{\mathsf{x},\mathsf{y},\mathsf{o},\mathsf{n} \in dom(l) \qquad l(\mathsf{y}) \in OId \qquad v \in CVal}{(l, inl(\mathsf{x} = \mathtt{CAS}(\mathsf{y}.\mathsf{f},\mathsf{o},\mathsf{n}))) \xrightarrow{cas(l(\mathsf{y}),\mathsf{f},l(\mathsf{o}),l(\mathsf{n}),v)} (l[\mathsf{x} \mapsto v], \varepsilon)}\ \textsc{CAS}$$

$$\frac{\mathsf{x},\mathsf{y} \in dom(l) \qquad l(\mathsf{x}) = l(\mathsf{y})}{(l, inl(\mathtt{if}\ (\mathsf{x}\ \mathtt{==}\ \mathsf{y})\ \{\mathsf{s}_1\}\ \mathtt{else}\ \{\mathsf{s}_2\})) \xrightarrow{id} (l, stm(\mathsf{s}_1))}\ \textsc{IfT}$$

$$\frac{\mathsf{x},\mathsf{y} \in dom(l) \qquad l(\mathsf{x}) \neq l(\mathsf{y})}{(l, inl(\mathtt{if}\ (\mathsf{x}\ \mathtt{==}\ \mathsf{y})\ \{\mathsf{s}_1\}\ \mathtt{else}\ \{\mathsf{s}_2\})) \xrightarrow{id} (l, stm(\mathsf{s}_2))}\ \textsc{IfF}$$

$$\frac{\mathsf{x} \in dom(l) \qquad o \in OId}{(l, inl(\mathsf{x} = \mathtt{new}\ T)) \xrightarrow{oalloc(T,o)} (l[\mathsf{x} \mapsto o], \varepsilon)}\ \textsc{CAlloc}$$

$$\frac{\mathsf{x},\mathsf{y} \in dom(l) \qquad l(\mathsf{y}) \in OId \qquad c \in CId}{(l, inl(\mathsf{x} = \mathtt{delegate}\ \mathsf{y}.\mathsf{m})) \xrightarrow{calloc(l(\mathsf{y}),\mathsf{m},c)} (l[\mathsf{x} \mapsto c], \varepsilon)}\ \textsc{DAlloc}$$

$$\frac{\begin{array}{c}\mathsf{x},\mathsf{y},\bar{\mathsf{z}} \in dom(l) \qquad l(\mathsf{y}) \in OId \qquad T \in CName \\ \mathsf{body}(T,\mathsf{m}) = \mathsf{C}\ \mathsf{m}(\overline{\mathsf{C}\mathsf{x}})\{\overline{C\mathsf{y}}; \bar{\mathsf{s}}_2; \mathtt{return}\ \mathsf{r}\}\end{array}}{(l, inl(\mathsf{x} = \mathsf{y}.\mathsf{m}(\bar{\mathsf{z}}))) \xrightarrow{otype(l(\mathsf{y}),T)} ([\mathtt{this} \mapsto l(\mathsf{y}), \bar{\mathsf{x}} \mapsto l(\bar{\mathsf{z}}), \bar{\mathsf{y}} \mapsto null], stm(\bar{\mathsf{s}}_2); return\ (l,\mathsf{x},\mathsf{r}))}\ \textsc{MCall}$$

$$\frac{\begin{array}{c}\mathsf{x},\mathsf{y},\bar{\mathsf{z}} \in dom(l) \qquad l(\mathsf{y}) \in CId \qquad T \in CName \qquad o \in OId \\ \mathsf{body}(T,\mathsf{m}) = \mathsf{C}\ \mathsf{m}(\overline{\mathsf{C}\mathsf{x}})\{\overline{C\mathsf{y}}; \bar{\mathsf{s}}_2; \mathtt{return}\ \mathsf{r}\}\end{array}}{(l, inl(\mathsf{x} = \mathsf{y}(\bar{\mathsf{z}}))) \xrightarrow{ctype(l(\mathsf{y}),T,o,\mathsf{m})} ([\mathtt{this} \mapsto o, \bar{\mathsf{x}} \mapsto l(\bar{\mathsf{z}}), \bar{\mathsf{y}} \mapsto null], stm(\bar{\mathsf{s}}_2); return\ (l,\mathsf{x},\mathsf{r}))}\ \textsc{DCall}$$

$$\frac{\mathsf{x} \in dom(l_2) \qquad \mathsf{y} \in dom(l_1)}{(l_1, return\ (l_2,\mathsf{x},\mathsf{y})) \xrightarrow{id} (l_2[\mathsf{x} \mapsto l_1(\mathsf{y})], \varepsilon)}\ \textsc{Return}$$

All of the above rules require that the local stack variables involved exist on the stack and contain values in the right semantic domains. For instance, the Read rule requires that x and y exist on the stack and that x contains an object identifier. If these conditions are not met, read should fault, as expressed by the following rule.

$$\frac{\mathsf{x} \notin dom(l) \vee \mathsf{y} \notin dom(l) \vee l(\mathsf{x}) \notin OId}{(l, inl(\mathsf{y} = \mathsf{x}.\mathsf{f})) \xrightarrow{\lightning} (l, \varepsilon)}\ \textsc{ReadF}$$

The other fault cases are similar and have been omitted.

*Single-step semantics* $\boxed{\rightarrow \subseteq \mathit{Thread} \times \mathit{Act} \times \mathit{TPool}}$

$$\frac{(l, s) \xrightarrow{a} (l', s_1)}{(t, l, s \cdot s_2) \xrightarrow{a} \{(t, l', s_1; s_2)\}}\ \textsc{seq}$$

$$\frac{\begin{array}{c}\mathsf{x} \in \mathit{dom}(l) \qquad l(\mathsf{x}) \in \mathit{CId} \qquad T \in \mathit{CName} \qquad o \in \mathit{OId} \\ \mathsf{body}(T, \mathsf{m}) = \texttt{void m()}\{\overline{\mathsf{Cy}}; \bar{\mathsf{s}}_2; \texttt{return r}\} \qquad t_1 \neq t_2\end{array}}{(t_1, l, \mathit{inl}(\texttt{fork}(\mathsf{x})) \cdot s_1) \xrightarrow{\mathit{ctype}(l(\mathsf{x}), T, o, \mathsf{m})} \{(t_1, l, s_1), (t_2, [\texttt{this} \mapsto o, \bar{\mathsf{y}} \mapsto \texttt{null}], \mathit{stm}(\bar{\mathsf{s}}_2))\}}\ \textsc{Fork}$$

There is a corresponding fault rule for `fork`, in case the stack variables involved are not defined or contain values in the wrong semantic domains or the delegate refers to a method that does not exist.

*Thread pool evaluation* $\boxed{\rightarrow \subseteq \mathit{TPool} \times \mathit{Act} \times \mathit{TPool}}$

$$\frac{x \in T \qquad x \xrightarrow{a} T' \qquad \mathit{utid}((T \setminus \{x\}) \cup T')}{T \xrightarrow{a} (T \setminus \{x\}) \cup T'}$$

*Single-step program evaluation* $\boxed{\rightarrow \subseteq \mathit{Prg} \times \mathit{Prg}}$

$$\frac{T \xrightarrow{a} T' \qquad h' \in [\![a]\!](h)}{(h, T) \rightarrow (h', T')}\ \textsc{Step}$$

*Irreducibility* $\boxed{\mathit{irr} \in \mathit{Thread} \rightarrow 2, \mathit{irr} \in \mathit{TPool} \rightarrow 2}$

$$\begin{array}{ll} \mathit{irr}(x) \stackrel{\text{def}}{=} \forall a \in \mathit{Act}.\ \forall T \in \mathit{TPool}.\ x \not\xrightarrow{a} T & \text{for } x \in \mathit{Thread} \\ \mathit{irr}(T) \stackrel{\text{def}}{=} \forall x \in T.\ \mathit{irr}(x) & \text{for } T \in \mathit{TPool} \end{array}$$

*Action semantics* $\boxed{[\![-]\!] : \mathit{Act} \times \mathit{Heap} \to \mathcal{P}(\mathit{Heap} \uplus \{\lightning\})}$

$$[\![id]\!](h) = \{h\}$$

$$[\![\lightning]\!](h) = \{\lightning\}$$

$$[\![read(o,f,v)]\!](h) = \begin{cases} \{h\} & \text{if } (o,f) \in dom(h) \text{ and } h(o,f) = v \\ \emptyset & \text{if } (o,f) \in dom(h) \text{ and } h(o,f) \neq v \\ \{\lightning\} & \text{if } (o,f) \notin dom(h) \end{cases}$$

$$[\![write(o,f,v)]\!](h) = \begin{cases} \{h[(o,\mathsf{f}) \mapsto v]\} & \text{if } (o,f) \in dom(h) \\ \{\lightning\} & \text{if } (o,f) \notin dom(h) \end{cases}$$

$$[\![cas(o,f,v_o,v_n,r)]\!](h) = \begin{cases} \{h\} & \text{if } (o,f) \in dom(h),\ h(o,f) \neq v_o \text{ and } r = null \\ \{h[(o,f) \mapsto v_n]\} & \text{if } (o,f) \in dom(h),\ h(o,f) = v_o \text{ and } r = o \\ \{\lightning\} & \text{if } (o,f) \notin dom(h) \end{cases}$$

$$[\![oalloc(T,o)]\!](h) = \begin{cases} \{h[(o,\bar{\mathsf{f}}) \mapsto null, o \mapsto T]\} & \text{if } o \notin dom(h) \text{ and } \mathsf{fields}(T) = \bar{\mathsf{f}} \\ \emptyset & \text{if } o \in dom(h) \\ \{\lightning\} & \text{if } \mathsf{fields}(T) \text{ is undefined} \end{cases}$$

$$[\![calloc(o,m,c)]\!](h) = \begin{cases} \{h[c \mapsto (o,m)]\} & \text{if } c \notin dom(h) \\ \emptyset & \text{if } c \in dom(h) \end{cases}$$

$$[\![otype(o,T)]\!](h) = \begin{cases} \{h\} & \text{if } o \in dom(h_t) \text{ and } h_t(o) = T \\ \emptyset & \text{if } o \in dom(h_t) \text{ and } h_t(o) \neq T \\ \{\lightning\} & \text{if } o \notin dom(h_t) \end{cases}$$

$$[\![ctype(c,T,o,m)]\!](h) = \begin{cases} \{h\} & \text{if } c \in dom(h),\ o \in dom(h_t),\ h(c) = (o,m) \text{ and } h_t(o) = T \\ \emptyset & \text{if } c \in dom(h),\ o \in dom(h_t), \text{ and } h(c) \neq (o,m) \text{ or } h_t(o) \neq T \\ \{\lightning\} & \text{if } c \notin dom(h) \text{ or } o \notin dom(h_t) \end{cases}$$

*Action semantics* $\boxed{[\![-]\!] : \mathit{Act} \times (\mathit{Heap} \uplus \{\lightning\}) \to \mathcal{P}(\mathit{Heap} \uplus \{\lightning\})}$

$$[\![a]\!](x) = \begin{cases} [\![a]\!](x) & \text{if } x \in \mathit{Heap} \\ \{\lightning\} & \text{if } x = \lightning \end{cases}$$

*Modifies set*

$$\begin{array}{l} \mathsf{mod}_1 : \mathit{Stm} \to \mathcal{P}(\mathit{Var}) \\ \mathsf{mod} : \mathit{seq}\ \mathit{Stm} \to \mathcal{P}(\mathit{Var}) \end{array}$$

$$\begin{aligned}
\mathsf{mod}_1(\mathsf{x} = \mathsf{y}) &= \{\mathsf{x}\} \\
\mathsf{mod}_1(\mathsf{x} = \mathsf{null}) &= \{\mathsf{x}\} \\
\mathsf{mod}_1(\mathsf{x} = \mathsf{y.f}) &= \{\mathsf{x}\} \\
\mathsf{mod}_1(\mathsf{x} = \texttt{new C}) &= \{\mathsf{x}\} \\
\mathsf{mod}_1(\mathsf{x} = \texttt{delegate y.m}) &= \{\mathsf{x}\} \\
\mathsf{mod}_1(\mathsf{x} = \texttt{CAS}(\mathsf{y.f}, \mathsf{o}, \mathsf{n})) &= \{\mathsf{x}\} \\
\mathsf{mod}_1(\mathsf{x} = \mathsf{y.m}(\bar{\mathsf{z}})) &= \{\mathsf{x}\} \\
\mathsf{mod}_1(\mathsf{x} = \mathsf{y}(\bar{\mathsf{z}})) &= \{\mathsf{x}\} \\
\mathsf{mod}_1(\mathsf{x.f} = \mathsf{y}) &= \emptyset \\
\mathsf{mod}_1(\texttt{fork}(\mathsf{x})) &= \emptyset \\
\mathsf{mod}_1(\texttt{if } (\mathsf{x} == \mathsf{y})\ \{\bar{\mathsf{s}}_1\}\ \texttt{else}\ \{\bar{\mathsf{s}}_2\}) &= \mathsf{mod}(\bar{\mathsf{s}}_1) \cup \mathsf{mod}(\bar{\mathsf{s}}_2)
\end{aligned}$$

$$\begin{aligned}
\mathsf{mod}(\varepsilon) &= \emptyset \\
\mathsf{mod}(\mathsf{s}_1; \bar{\mathsf{s}}_2) &= \mathsf{mod}_1(\mathsf{s}_1) \cup \mathsf{mod}(\bar{\mathsf{s}}_2)
\end{aligned}$$

## References


[1] L. Birkedal, R. Møgelberg, J. Schwinghammer, and K. Støvring. First steps in synthetic guarded domain theory: step-indexing in the topos of trees. *Logical Methods in Computer Science*, 8(4), Oct. 2012.

[2] T. Dinsdale-Young, L. Birkedal, P. Gardner, M. Parkinson, and H. Yang. Views: Compositional Reasoning for Concurrent Programs. In *Proceedings of POPL*, 2013.