Higher-order Concurrent Abstract Predicates

Kasper Svendsen
IT University of Copenhagen
kasv@itu.dk

Lars Birkedal
IT University of Copenhagen
birkedal@itu.dk

Matthew Parkinson
Microsoft Research, Cambridge
mattpark@microsoft.com

October 19, 2012

Contents

1 Mini C$^\sharp$

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

1.1 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	$::=$	$\texttt{class}\ \mathsf{C}\ \{\overline{\mathsf{Cf}}; \bar{\mathsf{M}}\}$	Class definition
M	$::=$	$\mathsf{C}\ \mathsf{m}(\overline{\mathsf{Cx}})\ \{\overline{\mathsf{Cy}}; \bar{\mathsf{s}}; \texttt{return}\ \mathsf{z}\}$	Method definition
s	$::=$		Statement
		`x = y`	assignment
		`x = null`	initialization
		`x = y.f`	field access
		`x.f = y`	field update
		$\texttt{if (x == y)}\ \{\bar{\mathsf{s}}_1\}\ \texttt{else}\ \{\bar{\mathsf{s}}_2\}$	conditional
		`x = new C()`	object creation
		`x = delegate y.m`	named delegate
		$\texttt{x = y.m}(\bar{\mathsf{z}})$	method invocation
		$\texttt{x = y}(\bar{\mathsf{z}})$	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.

1.2 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*).

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

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

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

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

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 $\mathit{return}\ (l, x, y)$ as notation for method returns. Formally, $\mathit{return}\ (l, x, y)$ is notation for $\mathit{inr}(l, x, y) \in \mathit{Stm} + (\mathit{Stack} \times \mathit{Var} \times \mathit{Var})$. We use $\mathit{stm}(s)$ as notation for $\mathit{map}(\mathit{inl})(s) \in \mathit{TCStack}$ when $s \in \mathit{Stm}$. 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 [3], and factors the small-step program evaluation relation, $\rightarrow$, into a labelled thread pool evaluation relation, $\overset{a}{\rightarrow}$, and an action semantics, $[\![-]\!]$. The labelled thread pool evaluation relation, $\overset{a}{\rightarrow}$, 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 4. 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 $\boxed{Act \in \text{Sets}}$

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

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

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

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

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

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

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

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

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

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

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

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

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

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.

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

The other fault cases are similar and have been omitted.

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

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

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

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 $\boxed{\to\ \subseteq\ \mathit{TPool} \times \mathit{Act} \times \mathit{TPool}}$

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

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

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

Multi-step program evaluation $\boxed{\to^n\ \subseteq\ \mathit{Prg} \times \mathit{Prg}}$

$$\frac{}{(h, T) \to^0 (h, T)} \qquad\qquad \frac{(h, T) \to (h'', T'') \qquad (h'', T'') \to^n (h', T')}{(h, T) \to^{n+1} (h', T')}$$

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

$$\mathit{irr}(x) \stackrel{\text{def}}{=} \forall a \in \mathit{Act}.\ \forall T \in \mathit{TPool}.\ x \stackrel{a}{\not\to} T \qquad \text{for } x \in \mathit{Thread}$$

$$\mathit{irr}(T) \stackrel{\text{def}}{=} \forall x \in T.\ \mathit{irr}(x) \qquad \text{for } T \in \mathit{TPool}$$

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

$$
\begin{aligned}
[\![\mathit{id}]\!](h) &= \{h\} \\
[\![\lightning]\!](h) &= \{\lightning\} \\
[\![\mathit{read}(o,f,v)]\!](h) &= \begin{cases} \{h\} & \text{if } (o,f) \in \mathit{dom}(h) \text{ and } h(o,f) = v \\ \emptyset & \text{if } (o,f) \in \mathit{dom}(h) \text{ and } h(o,f) \neq v \\ \{\lightning\} & \text{if } (o,f) \notin \mathit{dom}(h) \end{cases} \\
[\![\mathit{write}(o,f,v)]\!](h) &= \begin{cases} \{h[(o,\mathsf{f}) \mapsto v]\} & \text{if } (o,f) \in \mathit{dom}(h) \\ \{\lightning\} & \text{if } (o,f) \notin \mathit{dom}(h) \end{cases} \\
[\![\mathit{cas}(o,f,v_o,v_n,r)]\!](h) &= \begin{cases} \{h\} & \text{if } (o,f) \in \mathit{dom}(h),\ h(o,f) \neq v_o \text{ and } r = \mathit{null} \\ \{h[(o,f) \mapsto v_n]\} & \text{if } (o,f) \in \mathit{dom}(h),\ h(o,f) = v_o \text{ and } r = o \\ \{\lightning\} & \text{if } (o,f) \notin \mathit{dom}(h) \end{cases} \\
[\![\mathit{oalloc}(T,o)]\!](h) &= \begin{cases} \{h[(o,\bar{\mathsf{f}}) \mapsto \mathit{null}, o \mapsto T]\} & \text{if } o \notin \mathit{dom}(h) \text{ and } \mathsf{fields}(T) = \bar{\mathsf{f}} \\ \emptyset & \text{if } o \in \mathit{dom}(h) \\ \{\lightning\} & \text{if } \mathsf{fields}(T) \text{ is undefined} \end{cases} \\
[\![\mathit{calloc}(o,m,c)]\!](h) &= \begin{cases} \{h[c \mapsto (o,m)]\} & \text{if } c \notin \mathit{dom}(h) \\ \emptyset & \text{if } c \in \mathit{dom}(h) \end{cases} \\
[\![\mathit{otype}(o,T)]\!](h) &= \begin{cases} \{h\} & \text{if } o \in \mathit{dom}(h_t) \text{ and } h_t(o) = T \\ \emptyset & \text{if } o \in \mathit{dom}(h_t) \text{ and } h_t(o) \neq T \\ \{\lightning\} & \text{if } o \notin \mathit{dom}(h_t) \end{cases} \\
[\![\mathit{ctype}(c,T,o,m)]\!](h) &= \begin{cases} \{h\} & \text{if } c \in \mathit{dom}(h),\ o \in \mathit{dom}(h_t),\ h(c) = (o,m) \text{ and } h_t(o) = T \\ \emptyset & \text{if } c \in \mathit{dom}(h),\ o \in \mathit{dom}(h_t), \text{ and } h(c) \neq (o,m) \text{ or } h_t(o) \neq T \\ \{\lightning\} & \text{if } c \notin \mathit{dom}(h) \text{ or } o \notin \mathit{dom}(h_t) \end{cases}
\end{aligned}
$$

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

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

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

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

2 Logic

In this section we define a proof system for reasoning about mini C$^\sharp$ programs. The proof system consists of several components.

- A higher-order separation logic for reasoning about mutable data structures. The main ingredient is a separating conjunction connective, written $p * q$, which asserts that p and q holds disjointly.
- A higher-order variant of concurrent abstract predicates for reasoning about shared mutable data structures. Concurrent abstract predicates partitions the state into regions with protocols to describe how the state in each region is allowed to evolve. This allows local reasoning about shared mutable data structures, by providing an abstraction of possible environment interference.
- A higher-order specification logic for modular reasoning about libraries. This allows libraries to be specified abstractly, by existentially quantifying over representation predicates in library specifications. Since representation predicates describe both the state and possible interference, this allows specifications to abstract both the internal data-representation and any internal parallelism.
- An embedding from assertions into specifications, to allow specifications to expose axioms about representation predicates to clients. An embedding from specifications into assertions, to allow reasoning about delegates.
- A later modality and guarded recursion for reasoning about mutually recursive methods and recursion through the store. The later modality internalizes a notion of an execution step into the proof system. This allows reasoning about mutually recursive methods, by induction on execution steps.
- Phantom fields to record an abstraction of the state and history of execution.

Combined into one proof system, these features allow us to reason abstractly about libraries and clients that combine shared mutable data structures, concurrency, and recursion through the store. For a realistic example that combines all these features, see our work on the Joins library [6, 7].

2.1 Syntax

Conceptually, the program logic consists of two layers: an assertion logic for reasoning about program states and a specification logic for reasoning about programs. Since the language features delegates, we allow specifications to be embedded in assertions. Likewise, to allow library specifications to expose axioms about representation predicates to clients, we also allow assertions to be embedded in specifications. Formally, the proof system thus contains a single term language, defined below, containing both specifications and assertions.

$$\begin{array}{llll}
\text{Terms} & \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_F} \mapsto \mathsf{N} \mid \mathsf{M} \mapsto \mathsf{N.M} \mid \mathsf{M} : \mathsf{N} \\
& & \mid & C \mid \mathsf{m} \mid \mathsf{f} \mid \mathsf{null} \\
& & \mid & \varepsilon \mid \mathsf{M} \leq \mathsf{N} \mid \mathsf{M} \sqcap \mathsf{N} \\
& & \mid & \boxed{\mathsf{P}}^{\mathsf{R,T,M}} \mid \mathsf{protocol}(\mathsf{R,M,N}) \mid [\mathsf{M}]^{\mathsf{R}}_{\mathsf{N}} \mid p \\
& & \mid & \mathsf{pure_{protocol}}(\mathsf{P}) \mid \mathsf{pure_{state}}(\mathsf{P}) \mid \mathsf{pure_{perm}}(\mathsf{P}) \\
& & \mid & \mathsf{dep_T}(\mathsf{P}) \mid \mathsf{indep_T}(\mathsf{P}) \mid \mathsf{stable}(\mathsf{P}) \mid \mathsf{stable}^{\mathsf{R}}_{\mathsf{A}}(\mathsf{P}) \\
& & \mid & \mathsf{P} \sqsubseteq \mathsf{Q} \mid \mathsf{P} \sqsubseteq^{\mathsf{T}} \mathsf{Q} \mid (\Delta).\{\mathsf{P}\}\langle \mathsf{s}\rangle\{\mathsf{Q}\} \mid (\Delta).\{\mathsf{P}\}\langle \mathsf{s}\rangle^{\mathsf{T}}\{\mathsf{Q}\} \\
& & \mid & \mathsf{P} \rightsquigarrow^{\mathsf{N,M}} \mathsf{Q} \mid \mathsf{P} \rightsquigarrow^{\mathsf{N,M}}_{(\Delta).\langle \mathsf{s}\rangle} \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{asn}(\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}\}
\end{array}$$

where p is a fraction in $(0, 1]$.

We use a type system to carve out specifications and assertions from this common term language. The set of types is generated by the following grammar:

$$\begin{array}{llll}
\text{Types} & \tau, \sigma & ::= & 1 \mid \tau \to \sigma \mid \tau \times \sigma \mid \mathsf{Prop} \mid \mathsf{Spec} \\
& & \mid & \mathsf{Val} \mid \mathsf{Class} \mid \mathsf{Method} \mid \mathsf{Field} \\
& & \mid & \mathsf{Perm} \mid \mathsf{Action} \mid \mathsf{Region} \mid \mathsf{RType}
\end{array}$$

and includes the standard type constructors for a simply-typed lambda calculus. Basic types include the type of assertions, Prop, the type of specifications, Spec, and the type of mathematical values, Val. The Val type includes all C$^\sharp$ values and strings, and is closed under formation of pairs, such that mathematical sequences and other mathematical objects can be conveniently represented.[2] In addition, basic types include Class, Method, and Field, the types of C$^\sharp$ classes, methods and fields, respectively. Finally, basic types include the type of permissions, Perm, the type of action identifiers, Action, the type of region identifiers, Region, and the type of region types, RType.

As a convention, we mostly use meta-variables P and Q for terms of type Prop, S and T for terms of type Spec, C for terms of type Class, and F for terms of type Field. We also use P and T for terms of type Perm and RType, respectively.

[2]We use a single universe Val for the universe of mathematical values to avoid also having to quantify over types in the logic. We omit isomorphisms $\mathsf{Val} \cong \mathsf{Val} \times \mathsf{Val}$ and write $(v_1, ..., v_n)$ for tuples coded as elements of Val.

2.2 Typing

Terms are typed with the typing judgment $\Gamma; \Delta \vdash \mathsf{M} : \tau$, where Γ is a logical variable context and Δ a program variable context, and τ is a well-formed type. Well-formed types are given by the $\vdash \tau : \mathsf{Type}$ judgment. The type system thus consists of the following judgments:

$\vdash \tau : \mathsf{Type}$	τ is a well-formed type
$\Gamma; \Delta \vdash \mathsf{M} : \tau$	M is a well-formed term of type τ

where

$$\begin{array}{lcll} \Gamma & ::= & \Gamma, \mathsf{x} : \tau \mid \varepsilon & \text{logical variable context} \\ \Delta & ::= & \Delta, \mathsf{x} : \mathsf{Val} \mid \varepsilon & \text{program variable context} \end{array}$$

Specifications do not have any free program variables and are thus typed with an empty program variable context, Δ. Hence, we use $\Gamma \vdash \mathsf{M} : \tau$ as shorthand for $\Gamma; - \vdash \mathsf{M} : \tau$. We write Γ, Δ for the concatenation of Γ and Δ. Thus, $\Gamma, \Delta \vdash \mathsf{M} : \tau$ is shorthand for $\Gamma, \Delta; - \vdash \mathsf{M} : \tau$.

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

$$\frac{\vdash \tau : \mathsf{Type} \qquad \vdash \sigma : \mathsf{Type}}{\vdash \tau \times \sigma : \mathsf{Type}} \qquad \frac{\vdash \tau : \mathsf{Type} \qquad \vdash \sigma : \mathsf{Type}}{\vdash \tau \to \sigma : \mathsf{Type}}$$

$$\frac{}{\vdash 1 : \mathsf{Type}} \qquad \frac{}{\vdash \mathsf{Prop} : \mathsf{Type}} \qquad \frac{}{\vdash \mathsf{Spec} : \mathsf{Type}}$$

$$\frac{}{\vdash \mathsf{Class} : \mathsf{Type}} \qquad \frac{}{\vdash \mathsf{Method} : \mathsf{Type}} \qquad \frac{}{\vdash \mathsf{Field} : \mathsf{Type}} \qquad \frac{}{\vdash \mathsf{Val} : \mathsf{Type}}$$

$$\frac{}{\vdash \mathsf{Perm} : \mathsf{Type}} \qquad \frac{}{\vdash \mathsf{Action} : \mathsf{Type}} \qquad \frac{}{\vdash \mathsf{Region} : \mathsf{Type}} \qquad \frac{}{\vdash \mathsf{RType} : \mathsf{Type}}$$

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 x in the logic, and italic identifiers, such as x in the meta-logic.

Assertion typing

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

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

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

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

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

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

$$\frac{}{\Gamma; \Delta \vdash \mathsf{C} : \mathsf{Class}} \qquad \frac{}{\Gamma; \Delta \vdash \mathsf{m} : \mathsf{Method}} \qquad \frac{}{\Gamma; \Delta \vdash \mathsf{f} : \mathsf{Field}}$$

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

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

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

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

Specification & assertion embedding

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

The embedding of specifications into the assertion logic allows us to *define* nested Hoare triples [4] for delegates, by embedding a specification of the named method the delegate refers to.

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

This asserts that x refers to a named method m on object y, that object y has dynamic type C, and that the method m from the C class satisfies the given Hoare specification.

Guarded recursion typing

$$\frac{\Gamma; \Delta \vdash \mathsf{M} : (\tau \to \mathsf{Prop}) \to (\tau \to \mathsf{Prop})}{\Gamma; \Delta \vdash \mathsf{fix}_\tau(\mathsf{M}) : \tau \to \mathsf{Prop}} \qquad \frac{\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop}}{\Gamma; \Delta \vdash \triangleright \mathsf{P} : \mathsf{Prop}}$$

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

We omit the type subscript from fix and guarded when it is clear from the context.

Region types typing

$$\frac{}{\Gamma; \Delta \vdash \varepsilon : \mathsf{RType}} \qquad \frac{\Gamma; \Delta \vdash \mathsf{M} : \mathsf{RType} \qquad \Gamma; \Delta \vdash \mathsf{N} : \mathsf{RType}}{\Gamma; \Delta \vdash \mathsf{M} \leq \mathsf{N} : \mathsf{Prop}}$$

$$\frac{\Gamma; \Delta \vdash \mathsf{M} : \mathsf{RType} \qquad \Gamma; \Delta \vdash \mathsf{N} : \mathsf{RType}}{\Gamma; \Delta \vdash \mathsf{M} \sqcap \mathsf{N} : \mathsf{RType}}$$

Concurrent abstract predicates typing

$$\frac{\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma; \Delta \vdash \mathsf{R} : \mathsf{Region} \qquad \Gamma; \Delta \vdash \mathsf{T} : \mathsf{RType} \qquad \Gamma; \Delta \vdash \mathsf{A} : \mathsf{Val}}{\Gamma; \Delta \vdash \boxed{\mathsf{P}}^{\mathsf{R},\mathsf{T},\mathsf{A}} : \mathsf{Prop}}$$

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

$$\frac{\Gamma; \Delta \vdash \mathsf{R} : \mathsf{Region} \qquad \Gamma; \Delta \vdash \mathsf{I}_p, \mathsf{I}_q : \mathsf{Val} \times \mathsf{Action} \times \mathsf{Val} \to \mathsf{Prop}}{\Gamma; \Delta \vdash \mathsf{protocol}(\mathsf{R}, \mathsf{I}_p, \mathsf{I}_q) : \mathsf{Prop}}$$

Formally, protocols are given by a pair of parameterized action pre- and post-conditions I_p and I_q. Formally, these are given as predicates on a protocol argument of type Val, an action identifier of type Action and a action parameter of type Val.

We use the following informal notation for a parametric protocol I with parameter a and actions α_1, ..., α_n:

$$\mathsf{I}(\mathsf{a}) = \begin{pmatrix} \alpha_1 : (\mathsf{x}_1, ..., \mathsf{x}_k).\ p_1(\mathsf{a}, \mathsf{x}_1, ..., \mathsf{x}_k) \rightsquigarrow q_1(\mathsf{a}, \mathsf{x}_1, ..., \mathsf{x}_k) \\ \vdots \\ \alpha_n : (\mathsf{x}_1, ..., \mathsf{x}_k).\ p_n(\mathsf{a}, \mathsf{x}_1, ..., \mathsf{x}_k) \rightsquigarrow q_n(\mathsf{a}, \mathsf{x}_1, ..., \mathsf{x}_k) \end{pmatrix}$$

Here $(\mathsf{a}, \mathsf{x}_1, ..., \mathsf{x}_k)$ is a context of logical variables of type Val, relating the action pre-condition p_i with the action post-condition q_i. Formally, this corresponds to the protocol given by the following action pre- and post-conditions:

$$\mathsf{I}_p(a, \alpha, y) = \begin{cases} p_i(a, \pi_1(y), ..., \pi_k(y)) & \text{if } \alpha = \alpha_i \\ \bot & \text{if } \forall i.\ \alpha \neq \alpha_i \end{cases}$$

$$\mathsf{I}_q(a, \alpha, y) = \begin{cases} q_i(a, \pi_1(y), ..., \pi_k(y)) & \text{if } \alpha = \alpha_i \\ \bot & \text{if } \forall i.\ \alpha \neq \alpha_i \end{cases}$$

where π denotes the assumed projection operation on Val. Furthermore, we use $\boxed{\mathsf{P}}_{\mathsf{I}}^{r,t,(a_1,...,a_k)}$ as shorthand for

$$\boxed{\mathsf{P}}_{\mathsf{I}_p,\mathsf{I}_q}^{r,t,\langle a_1,...,a_k\rangle}$$

where $\langle - \rangle$ denotes the assumed pairing operator on Val and I_p and I_q are the formal action pre- and post-conditions induced by the parametric protocol I.

$$\frac{0 < p \leq 1}{\Gamma; \Delta \vdash p : \mathsf{Perm}} \qquad \frac{}{\Gamma; \Delta \vdash a : \mathsf{Action}}$$

$$\frac{\Gamma \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma \vdash \mathsf{Q} : \mathsf{Prop}}{\Gamma \vdash \mathsf{P} \sqsubseteq \mathsf{Q} : \mathsf{Spec}} \qquad \frac{\begin{array}{c}\Gamma \vdash \mathsf{T} : \mathsf{RType} \\ \Gamma \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma \vdash \mathsf{Q} : \mathsf{Prop}\end{array}}{\Gamma \vdash \mathsf{P} \sqsubseteq^{\mathsf{T}} \mathsf{Q} : \mathsf{Spec}}$$

$$\frac{\begin{array}{cc}\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop} & \Gamma; \Delta \vdash \mathsf{Q} : \mathsf{Prop} \\ \Gamma; \Delta \vdash \mathsf{T} : \mathsf{RType} & \Gamma; \Delta \vdash \mathsf{R} : \mathsf{Region}\end{array}}{\Gamma \vdash \mathsf{P} \rightsquigarrow^{\mathsf{R},\mathsf{T}}_{(\Delta).\langle \mathsf{s} \rangle} \mathsf{Q} : \mathsf{Spec}} \qquad \frac{\begin{array}{cc}\Gamma \vdash \mathsf{P} : \mathsf{Prop} & \Gamma \vdash \mathsf{Q} : \mathsf{Prop} \\ \Gamma \vdash \mathsf{T} : \mathsf{RType} & \Gamma \vdash \mathsf{R} : \mathsf{Region}\end{array}}{\Gamma \vdash \mathsf{P} \rightsquigarrow^{\mathsf{R},\mathsf{T}} \mathsf{Q} : \mathsf{Spec}}$$

$$\frac{\Gamma \vdash \mathsf{P} : \mathsf{Prop}}{\Gamma \vdash \mathsf{stable}(\mathsf{P}) : \mathsf{Spec}} \qquad \frac{\Gamma \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma \vdash \mathsf{R} : \mathsf{Region} \qquad \Gamma \vdash \mathsf{A} : \mathsf{Action}}{\Gamma \vdash \mathsf{stable}^{\mathsf{R}}_{\mathsf{A}}(\mathsf{P}) : \mathsf{Spec}}$$

$$\frac{\Gamma \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma \vdash \mathsf{T} : \mathsf{RType}}{\Gamma \vdash \mathsf{dep}_{\mathsf{T}}(\mathsf{P}) : \mathsf{Spec}} \qquad \frac{\Gamma \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma \vdash \mathsf{T} : \mathsf{RType}}{\Gamma \vdash \mathsf{indep}_{\mathsf{T}}(\mathsf{P}) : \mathsf{Spec}}$$

$$\frac{\Gamma \vdash \mathsf{P} : \mathsf{Prop}}{\Gamma \vdash \mathsf{pure}_{\mathsf{perm}}(\mathsf{P}) : \mathsf{Spec}} \qquad \frac{\Gamma \vdash \mathsf{P} : \mathsf{Prop}}{\Gamma \vdash \mathsf{pure}_{\mathsf{state}}(\mathsf{P}) : \mathsf{Spec}} \qquad \frac{\Gamma \vdash \mathsf{P} : \mathsf{Prop}}{\Gamma \vdash \mathsf{pure}_{\mathsf{protocol}}(\mathsf{P}) : \mathsf{Spec}}$$

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

The specification logic features five atomic propositions for Hoare-style partial correctness reasoning. The three basic Hoare-assertions (HoareS, HoareM and HoareC) are for specifying statements, methods and constructors, respectively. The proof rules for these assertions (See Section 2.3.5) enforce stability of the pre- and post-condition.

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

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

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

The specification logic features another two Hoare-assertions for specifying atomic statements. These rules specifically do *not* enforce stability of the pre- and post-condition.

$$\frac{\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma; \Delta \vdash \mathsf{Q} : \mathsf{Prop}}{\Gamma \vdash (\Delta).\{\mathsf{P}\}\langle \mathsf{s}\rangle\{\mathsf{Q}\} : \mathsf{Spec}}$$

$$\frac{\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma; \Delta \vdash \mathsf{Q} : \mathsf{Prop} \qquad \Gamma \vdash \mathsf{T} : \mathsf{RType}}{\Gamma \vdash (\Delta).\{\mathsf{P}\}\langle \mathsf{s}\rangle^{\mathsf{T}}\{\mathsf{Q}\} : \mathsf{Spec}}$$

2.3 Logics

The specification logic is given by the specification entailment judgment

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

where S is a specification and Φ 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 T in Φ.

The assertion logic is given by the assertion entailment judgment

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

where P and 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 T in Φ. The assertion entailment includes the specification context Φ, to allow the use of assertion assumptions embedded in specifications.

2.3.1 Assertion logic

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

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

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

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

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

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

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

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

2.3.2 Specification logic

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

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

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

$$\frac{\Gamma \mid \Phi \vdash S \wedge T}{\Gamma \mid \Phi \vdash S} \qquad \frac{\Gamma \mid \Phi \vdash S \wedge T}{\Gamma \mid \Phi \vdash T} \qquad \frac{\Gamma \mid \Phi \vdash S_1 \vee S_2 \qquad \Gamma \mid \Phi, S_i \vdash T}{\Gamma \mid \Phi \vdash T}$$

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

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

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

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

$$\frac{\Gamma \mid \Phi, S \vdash (\Delta).\{P\}\bar{s}\{Q\}}{\Gamma \mid \Phi \vdash (\Delta).\{P * \mathsf{asn}(S)\}\bar{s}\{Q\}}\ \textsc{AsnI} \qquad \frac{\Gamma \mid \Phi \vdash (\Delta).\{P * \mathsf{asn}(S)\}\bar{s}\{Q\}}{\Gamma \mid \Phi, S \vdash (\Delta).\{P\}\bar{s}\{Q\}}\ \textsc{AsnE}$$

2.3.3 Guarded recursion and later modalities

The logic features a fixed-point operator fix on predicate functionals, which defines a fixed-point when applied to guarded predicate functionals.

$$\frac{\Gamma \vdash \mathsf{M} : (\tau \to \mathsf{Prop}) \to (\tau \to \mathsf{Prop}) \qquad \Gamma \mid \Phi \vdash \mathsf{guarded}(\mathsf{M})}{\Gamma \mid \Phi \vdash \forall \mathsf{x} : \tau.\ \mathsf{valid}(\mathsf{fix}(\mathsf{M})(\mathsf{x}) \Leftrightarrow \mathsf{M}(\mathsf{fix}(\mathsf{M}))(\mathsf{x}))}$$

Definitions are guarded using the later modality, $\triangleright$. To support modular guardedness proofs, we build non-expansiveness into the interpretation. A definition is thus guarded, if it can be expressed as a composition of functions where one function is guarded:

$$\frac{\Gamma \vdash \mathsf{M}_1, \mathsf{M}_2, \mathsf{M}_3 : (\tau \to \mathsf{Prop}) \to (\tau \to \mathsf{Prop}) \qquad \Gamma \mid \Phi \vdash \mathsf{guarded}(\mathsf{M}_2)}{\Gamma \mid \Phi \vdash \mathsf{guarded}(\mathsf{M}_3 \circ \mathsf{M}_2 \circ \mathsf{M}_1)}$$

where $\mathsf{M} \circ \mathsf{N} = \lambda P : \tau \to \mathsf{Prop}.\ \mathsf{M}(\mathsf{N}(P))$. Furthermore, recursive predicates are guarded if all occurrences of the recursive predicate appears under $\triangleright$ operators:

$$\frac{\Gamma \vdash \mathsf{M} : (\tau \to \mathsf{Prop}) \to (\tau \to \mathsf{Prop})}{\Gamma \mid \Phi \vdash \mathsf{guarded}(\lambda \mathsf{P} : \tau \to \mathsf{Prop}.\ \mathsf{M} \circ (\lambda \mathsf{x} : \tau.\ \triangleright \mathsf{P}(\mathsf{x})))}\ \textsc{guard}\mathrm{I}$$

$$\frac{\Gamma \vdash \mathsf{M} : (\tau \to \mathsf{Prop}) \to (\tau \to \mathsf{Prop})}{\Gamma \mid \Phi \vdash \mathsf{guarded}(\lambda \mathsf{P} : \tau \to \mathsf{Prop}.\ \lambda \mathsf{x} : \tau.\ \triangleright (\mathsf{M} \circ \mathsf{P})(\mathsf{x}))}\ \textsc{guard}\mathrm{E}$$

The later modality internalizes a notion of step in the logic. If the specification S holds for i steps of execution, then $\triangleright\mathsf{S}$ holds for $i+1$ steps of execution. This allows us to reason about mutually recursive methods and recursion through the store, using the LOEB rule, which internalizes induction on steps in the model. See Section 3.1 for an example that uses recursion through the store.

$$\frac{\Gamma \mid \Phi, \triangleright\mathsf{S} \vdash \mathsf{S}}{\Gamma \mid \Phi \vdash \mathsf{S}}\ \textsc{Loeb} \qquad\qquad \frac{}{\Gamma \mid \Phi, \mathsf{S} \vdash \triangleright\mathsf{S}}$$

The $\triangleright$-operator commutes with existentials over inhabited types (and every type is currently inhabited), and over separating conjunction.

$$\frac{\Gamma, \mathsf{x} : \tau \vdash \mathsf{P} : \mathsf{Prop} \qquad \mathit{inhabited}(\tau)}{\Gamma \mid \Phi \vdash \mathsf{valid}(\triangleright(\exists \mathsf{x} : \tau.\ \mathsf{P}) \Leftrightarrow (\exists \mathsf{x} : \tau.\ \triangleright \mathsf{P}))} \qquad \frac{\Gamma \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma \vdash \mathsf{Q} : \mathsf{Prop}}{\Gamma \mid \Phi \vdash \mathsf{valid}(\triangleright(\mathsf{P} * \mathsf{Q}) \Leftrightarrow (\triangleright\mathsf{P} * \triangleright\mathsf{Q}))}$$

2.3.4 Concurrent abstract predicates

The logic includes a large set of inference rules for reasoning about concurrent abstract predicates in the logic, without resorting to the semantics. This logic suffices for verifying realistic examples such as the spin-lock in Section 3.2 and the joins library (See [6] and accompanying technical report [7]). The logic for reasoning about concurrent abstract predicates consists of several parts:

- A logic for reasoning about stability. The logic allows stability proofs to be decomposed into stability under individual actions. The corner-stone of the stability logic is rule StableA, which allows one to prove stability of a region assertion containing nested (but not self-referential) region assertions.
- A logic for reasoning about accesses and updates to resources in a shared region. The main rule of this logic is OpenA, which allows a shared region to be "opened", moving the shared resources into the local state.
- A logic for reasoning about view-shifts on shared regions. Like the logic for reasoning about accesses and updates, the main rule, OpenV, allows a shared region to be "opened".
- A logic for reasoning about whether an atomic access or update to the resources of a shared region is allowed by the protocol on the given region.
- A logic for reasoning about whether a view-shift on the resources of a shared region is allowed by the protocol on the given region.
- A logic for reasoning about the region support of an assertion, to prove the absence of self-referential region and protocol assertions.
- Logics for reasoning about whether assertions makes assertions about protocols and state, respectively, to prove that assertions are expressible using state-independent protocols.

Stability

Stability is closed under all the standard assertion connectives and quantifiers.

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

$$\frac{\Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{P}) \qquad \Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{Q}) \qquad op \in \{\vee, \wedge, \Rightarrow, *\}}{\Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{P}\ op\ \mathsf{Q})}$$

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

Stability decomposes into stability under every action, considered individually.

$$\frac{\Gamma \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma \vdash \mathsf{R} : \mathsf{Region}}{\Gamma \mid \Phi \vdash (\forall \alpha : \mathsf{Action}.\ \mathsf{stable}_{\alpha}^{\mathsf{R}}(\mathsf{P})) \Leftrightarrow \mathsf{stable}(\mathsf{P})}\ \textsc{StableD}$$

Here the $\mathsf{stable}^{\mathsf{R}}_{\alpha}(\mathsf{P})$ specification expresses that the assertion P is stable under arbitrary actions on any region other than R and the α action on region R (See definition $\mathit{stable}^{r,A}$ on page 75 for the formal semantics of $\mathsf{stable}^{\mathsf{R}}_{\alpha}$). A proposition P is thus stable if there *exists* an R such that $\mathsf{stable}^{\mathsf{R}}_{\alpha}(\mathsf{P})$ holds for all actions α.

An assertion is stable under actions that it owns (StableOwn). Furthermore, a region assertion is stable under an action if it is closed under every instantiation of the action that satisfies the pre-condition (StableClosed).

$$\frac{\begin{array}{c}\Gamma \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma \vdash \mathsf{R} : \mathsf{Region} \\ \Gamma \vdash \mathsf{T} : \mathsf{RType} \qquad \Gamma \vdash \mathsf{A} : \mathsf{Val} \qquad \Gamma \vdash \mathsf{I}_p, \mathsf{I}_q : \mathsf{Val} \times \mathsf{Action} \times \mathsf{Val} \to \mathsf{Prop}\end{array}}{\Gamma \mid \Phi \vdash \mathsf{stable}^{\mathsf{R}}_{\alpha}\left(\boxed{\mathsf{P}}^{\mathsf{R,T,A}}_{\mathsf{I}_p,\mathsf{I}_q} * [\alpha]^{\mathsf{R}}_{1}\right)} \textsc{StableOwn}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathsf{R} : \mathsf{Region} \qquad \Gamma \vdash \mathsf{T} : \mathsf{RType} \qquad \Gamma \vdash \mathsf{A} : \mathsf{Val} \qquad \Gamma \vdash \alpha : \mathsf{Action} \\ \Gamma \vdash \mathsf{P}, \mathsf{Q} : \mathsf{Prop} \qquad \Gamma \vdash \mathsf{I}_p, \mathsf{I}_q : \mathsf{Val} \times \mathsf{Action} \times \mathsf{Val} \to \mathsf{Prop} \\ \Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{protocol}}(\mathsf{P}) \qquad \Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{state}}(\mathsf{Q}) \\ \Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{P} * \mathsf{Q}) \qquad \Gamma \mid \Phi \vdash \mathsf{indep}_{\mathsf{T}}(\mathsf{P}) \\ \Gamma \mid \Phi \vdash \forall \mathsf{x} : \mathsf{Val}.\ \mathsf{valid}((\mathsf{I}_p(\mathsf{A}, \alpha,\ \mathsf{x}) \wedge \mathsf{P}) \Rightarrow \bot) \vee \mathsf{valid}(\mathsf{I}_q(\mathsf{A}, \alpha, \mathsf{x}) \Rightarrow \mathsf{P})\end{array}}{\Gamma \mid \Phi \vdash \mathsf{stable}^{\mathsf{R}}_{\alpha}\left(\boxed{\mathsf{P}}^{\mathsf{R,T,A}}_{\mathsf{I}_p,\mathsf{I}_q} * \mathsf{Q}\right)} \textsc{StableClosed}$$

Since the underlying model lacks support for general higher-order protocols, the StableClosed rule requires that the region assertion can be expressed using state-independent protocols. We say an assertion R can be expressed using state-independent protocols if it can be written as $\mathsf{S} * \mathsf{P}$ where S makes no assertions about protocols and P makes no assertions about the state:

$$\mathsf{sip} = \lambda \mathsf{R} : \mathsf{Prop}.\ \exists \mathsf{S}, \mathsf{P} : \mathsf{Prop}.\ \mathsf{valid}(\mathsf{R} \Leftrightarrow \mathsf{S} * \mathsf{P}) \wedge \mathsf{pure}_{\mathsf{protocol}}(\mathsf{S}) \wedge \mathsf{pure}_{\mathsf{state}}(\mathsf{P})$$

Here $\mathsf{pure}_{\mathsf{protocol}}(\mathsf{S})$ asserts that S is invariant under arbitrary changes to protocols and $\mathsf{pure}_{\mathsf{state}}(\mathsf{P})$ asserts that P is invariant under arbitrary changes to the local and shared state.

For assertions that are expressible using state-independent protocols, protocol assertions can be "pulled outside" region assertions, to allow the StableClosed rule to be used. In particular, if $\mathsf{valid}(\mathsf{R} \Leftrightarrow (\mathsf{S} * \mathsf{P}))$, $\mathsf{pure}_{\mathsf{protocol}}(\mathsf{S})$ and $\mathsf{pure}_{\mathsf{state}}(\mathsf{P})$, then

$$\boxed{\mathsf{R}}^{\mathsf{R,T,A}}_{\mathsf{I}_p,\mathsf{I}_q} \Leftrightarrow \boxed{\mathsf{S}}^{\mathsf{R,T,A}}_{\mathsf{I}_p,\mathsf{I}_q} * \mathsf{P}$$

The left-hand side is thus stable if and only if the right-hand side is, and we can apply the StableClosed rule to the right-hand side.

$$\frac{\Gamma \mid \Phi, \mathsf{S} \vdash \mathsf{stable}(\mathsf{P}) \qquad \Gamma; - \mid \Phi \mid \mathsf{P} \vdash \mathsf{asn}(\mathsf{S})}{\Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{P})} \textsc{StableAsn}$$

When defining higher-order representation predicates, we will often use specification assertions to constrain instantiations of assertion and predicate arguments. The above rule allows us to copy such embedded specifications from an assertion P into the specification context, when proving stability of P.

View-shifts and atomic updates

$$\frac{\begin{array}{c}\Gamma \vdash \mathsf{R} : \mathsf{Region} \qquad \Gamma \vdash \mathsf{T}_1, \mathsf{T}_2 : \mathsf{RType} \qquad \Gamma \vdash \mathsf{A} : \mathsf{Val} \\ \Gamma \vdash \mathsf{P}_1, \mathsf{P}_2, \mathsf{Q}_1, \mathsf{Q}_2 : \tau \to \mathsf{Prop} \qquad \Gamma \mid \Phi \vdash \mathsf{T}_2 \not\leq \mathsf{T}_1 \\ \Gamma \mid \Phi \vdash \mathsf{indep}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}(\mathsf{P}_1) \qquad \Gamma \mid \Phi \vdash \mathsf{indep}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}(\mathsf{P}_2) \\ \Gamma \mid \Phi \vdash \mathsf{indep}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}(\mathsf{Q}_1) \qquad \Gamma \mid \Phi \vdash \mathsf{indep}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}(\mathsf{Q}_2) \\ \Gamma \mid \Phi \vdash \exists \mathsf{x} : \tau.\ \boxed{\mathsf{P}_1(\mathsf{x})}^{\mathsf{R},\mathsf{T}_1,\mathsf{A}} * \mathsf{P}_2(\mathsf{x}) \rightsquigarrow^{\mathsf{R},\mathsf{T}_2} \exists \mathsf{x} : \tau.\ \boxed{\mathsf{Q}_1(\mathsf{x})}^{\mathsf{R},\mathsf{T}_1,\mathsf{A}} * \mathsf{Q}_2(\mathsf{x}) \\ \Gamma; - \mid \Phi \mid \exists \mathsf{x} : \tau.\ \mathsf{P}_1(\mathsf{x}) * \mathsf{P}_2(\mathsf{x}) \vdash \exists \mathsf{x} : \tau.\ \mathsf{Q}_1(\mathsf{x}) * \mathsf{Q}_2(\mathsf{x})\end{array}}{\Gamma \mid \Phi \vdash \exists \mathsf{x} : \tau.\ \boxed{\mathsf{P}_1(\mathsf{x})}^{\mathsf{R},\mathsf{T}_1,\mathsf{A}} * \mathsf{P}_2(\mathsf{x}) \sqsubseteq^{\mathsf{T}_2} \exists \mathsf{x} : \tau.\ \boxed{\mathsf{Q}_1(\mathsf{x})}^{\mathsf{R},\mathsf{T}_1,\mathsf{A}} * \mathsf{Q}_2(\mathsf{x})}\ \textsc{OpenV}$$

To support modular stability proofs, action permissions owned by shared regions cannot be used to justify updates to shared regions. The following rule, which "opens" the shared region R thus explicitly requires that P_1 does not assert local ownership of any permissions. This ensures that no actions from P_1 are used to justify updates in the nested atomic update.

$$\frac{\begin{array}{c}\Gamma, \Delta \vdash \mathsf{R} : \mathsf{Region} \qquad \Gamma, \Delta \vdash \mathsf{T}_1, \mathsf{T}_2 : \mathsf{RType} \qquad \Gamma, \Delta \vdash \mathsf{A} : \mathsf{Val} \\ \Gamma, \Delta \vdash \mathsf{P}_1, \mathsf{P}_2, \mathsf{Q}_1, \mathsf{Q}_2 : \tau \to \mathsf{Prop} \\ \Gamma, \Delta \mid \Phi \vdash \mathsf{T}_2 \not\leq \mathsf{T}_1 \qquad \Gamma, \Delta \mid \Phi \vdash \mathsf{pure}_{\mathsf{perm}}(\mathsf{P}_1) \\ \Gamma, \Delta \mid \Phi \vdash \mathsf{indep}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}(\mathsf{P}_1) \qquad \Gamma, \Delta \mid \Phi \vdash \mathsf{indep}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}(\mathsf{P}_2) \\ \Gamma, \Delta \mid \Phi \vdash \mathsf{indep}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}(\mathsf{Q}_1) \qquad \Gamma, \Delta \mid \Phi \vdash \mathsf{indep}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}(\mathsf{Q}_2) \\ \Gamma \mid \Phi \vdash \exists \mathsf{x} : \tau.\ \boxed{\mathsf{P}_1(\mathsf{x})}^{\mathsf{R},\mathsf{T}_1,\mathsf{A}} * \mathsf{P}_2(\mathsf{x}) \rightsquigarrow^{\mathsf{R},\mathsf{T}_2}_{(\Delta).\langle s \rangle} \exists \mathsf{x} : \tau.\ \boxed{\mathsf{Q}_1(\mathsf{x})}^{\mathsf{R},\mathsf{T}_1,\mathsf{A}} * \mathsf{Q}_2(\mathsf{x}) \\ \Gamma \mid \Phi \vdash (\Delta).\{\exists \mathsf{x} : \tau.\ \mathsf{P}_1(\mathsf{x}) * \mathsf{P}_2(\mathsf{x})\} \langle s \rangle^{\mathsf{T}_1 \sqcap \mathsf{T}_2} \{\exists \mathsf{x} : \tau.\ \mathsf{Q}_1(\mathsf{x}) * \mathsf{Q}_2(\mathsf{x})\}\end{array}}{\Gamma \mid \Phi \vdash (\Delta).\{\exists \mathsf{x} : \tau.\ \boxed{\mathsf{P}_1(\mathsf{x})}^{\mathsf{R},\mathsf{T}_1,\mathsf{A}} * \mathsf{P}_2(\mathsf{x})\} \langle s \rangle^{\mathsf{T}_2} \{\exists \mathsf{x} : \tau.\ \boxed{\mathsf{Q}_1(\mathsf{x})}^{\mathsf{R},\mathsf{T}_1,\mathsf{A}} * \mathsf{Q}_2(\mathsf{x})\}}\ \textsc{OpenA}$$

$$\frac{\begin{array}{c}\Gamma, \Delta \vdash \mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3 : \mathsf{RType} \\ \Gamma, \Delta \mid \Phi \vdash (\mathsf{T}_1 = \mathsf{T}_2 = \varepsilon) \vee (\mathsf{T}_1 = \mathsf{T}_3 = \varepsilon) \vee (\mathsf{T}_2 = \mathsf{T}_3 = \varepsilon) \\ \Gamma, \Delta \mid \Phi \vdash \mathsf{P} \sqsubseteq^{\mathsf{T}_1} \mathsf{P}' \qquad \Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}'\} \langle s \rangle^{\mathsf{T}_2} \{\mathsf{Q}'\} \qquad \Gamma, \Delta \mid \Phi \vdash \mathsf{Q}' \sqsubseteq^{\mathsf{T}_3} \mathsf{Q}\end{array}}{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}\} \langle s \rangle^{\mathsf{T}_1 \sqcap \mathsf{T}_2 \sqcap \mathsf{T}_3} \{\mathsf{Q}\}}\ \textsc{ATrans}$$

$$\frac{\begin{array}{c}\Gamma, \Delta \mid \Phi \vdash \mathsf{T}_1 \leq \mathsf{T}_2 \\ \Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}\} \langle s \rangle^{\mathsf{T}_1} \{\mathsf{Q}\}\end{array}}{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}\} \langle s \rangle^{\mathsf{T}_2} \{\mathsf{Q}\}} \qquad \frac{\begin{array}{c}\Gamma \mid \Phi \vdash \mathsf{T}_1 \leq \mathsf{T}_2 \\ \Gamma \mid \Phi \vdash \mathsf{P} \sqsubseteq^{\mathsf{T}_1} \mathsf{Q}\end{array}}{\Gamma \mid \Phi \vdash \mathsf{P} \sqsubseteq^{\mathsf{T}_2} \mathsf{Q}}$$

$$\frac{\Gamma \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma \vdash \alpha_1, ..., \alpha_n : \mathsf{Action} \qquad \Gamma \vdash \mathsf{N} : \mathsf{Val} \qquad \Gamma \vdash \mathsf{T} : \mathsf{RType}}{\Gamma \mid \Phi \vdash \mathsf{P} \sqsubseteq^{\varepsilon} \exists \mathsf{R} : \mathsf{Region}.\ \boxed{\mathsf{P}}^{\mathsf{R},\mathsf{T},\mathsf{N}} * [\alpha_1]^{\mathsf{R}}_1 * \cdots * [\alpha_n]^{\mathsf{R}}_1}$$

$$\frac{\Gamma \vdash \mathsf{T}_1 : \mathsf{RType} \qquad \Gamma \vdash \mathsf{I}_p, \mathsf{I}_q : \mathsf{Val} \times \mathsf{Action} \times \mathsf{Val} \to \mathsf{Prop}}{\Gamma \mid \Phi \vdash \mathsf{emp} \sqsubseteq^{\varepsilon} \exists \mathsf{T}_2 : \mathsf{RType}.\ \mathsf{T}_1 \leq \mathsf{T}_2 * \mathsf{protocol}(\mathsf{T}_2, \mathsf{I}_p, \mathsf{I}_q)}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathsf{T} : \mathsf{RType} \qquad \Gamma \vdash \mathsf{P}_1, \mathsf{P}_2, \mathsf{Q} : \mathsf{Prop} \\ \Gamma \mid \Phi \vdash \mathsf{P}_1 \sqsubseteq^{\mathsf{T}} \mathsf{Q} \qquad \Gamma \mid \Phi \vdash \mathsf{P}_2 \sqsubseteq^{\mathsf{T}} \mathsf{Q}\end{array}}{\Gamma \mid \Phi \vdash \mathsf{P}_1 \vee \mathsf{P}_2 \sqsubseteq^{\mathsf{T}} \mathsf{Q}}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathsf{P}, \mathsf{Q}, \mathsf{R} : \mathsf{Prop} \\ \Gamma \mid \Phi \vdash \mathsf{P} \sqsubseteq \mathsf{Q} \qquad \Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{R})\end{array}}{\Gamma \mid \Phi \vdash \mathsf{P} * \mathsf{R} \sqsubseteq \mathsf{Q} * \mathsf{R}} \qquad \frac{\begin{array}{c}\Gamma \vdash \mathsf{T} : \mathsf{RType} \qquad \Gamma \vdash \mathsf{P}, \mathsf{Q}, \mathsf{R} : \mathsf{Prop} \\ \Gamma \mid \Phi \vdash \mathsf{P} \sqsubseteq^{\mathsf{T}} \mathsf{Q} \qquad \Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{R})\end{array}}{\Gamma \mid \Phi \vdash \mathsf{P} * \mathsf{R} \sqsubseteq^{\mathsf{T}} \mathsf{Q} * \mathsf{R}}$$

$$\frac{\Gamma; - \mid \Phi \mid \mathsf{P} \vdash \mathsf{Q}}{\Gamma \mid \Phi \vdash \mathsf{P} \sqsubseteq \mathsf{Q}}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathsf{T} : \mathsf{RType} \qquad \Gamma, \Delta \vdash \mathsf{P}, \mathsf{Q} : \mathsf{Prop} \\ \Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}\}\langle \mathsf{s} \rangle^{\mathsf{T}}\{\mathsf{Q}\} \qquad \mathsf{atomic}(\mathsf{s})\end{array}}{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}\}\mathsf{s}\{\mathsf{Q}\}} \qquad \frac{\begin{array}{c}\Gamma \vdash \mathsf{T} : \mathsf{RType} \\ \Gamma \vdash \mathsf{P}, \mathsf{Q} : \mathsf{Prop} \\ \Gamma \mid \Phi \vdash \mathsf{P} \sqsubseteq^{\mathsf{T}} \mathsf{Q}\end{array}}{\Gamma \mid \Phi \vdash \mathsf{P} \sqsubseteq \mathsf{Q}}$$

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

Atomic statements

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

Atomic update & view-shift allowed

The proof system includes two specifications assertions,

$$\mathsf{P} \rightsquigarrow^{\mathsf{R},\mathsf{T}} \mathsf{Q} \qquad \text{and} \qquad \mathsf{P} \rightsquigarrow^{\mathsf{R},\mathsf{T}}_{(\Delta).\langle \mathsf{s} \rangle} \mathsf{Q},$$

for asserting that a given view-shift/atomic update is permitted according to the protocol on region R. More formally, $\mathsf{P} \rightsquigarrow^{\mathsf{R},\mathsf{T}} \mathsf{Q}$ asserts that for any step at the instrumented level from P to Q that corresponds to a no-op at the concrete level, the update to region R is justified by the protocol on region R using an action owned by P. Furthermore, the action used to justify the update only depends on regions with region types greater than or equal to T. The meaning of $\mathsf{P} \rightsquigarrow^{\mathsf{R},\mathsf{T}}_{(\Delta).\langle \mathsf{s}\rangle} \mathsf{Q}$ is almost the same, as it asserts that any step at the instrumented level from P to Q that corresponds to the atomic statement s at the concrete level is permitted by region R. See the definition of $p \rightsquigarrow^{r,A}_{a} q$ and $p \rightsquigarrow^{r,A} q$ on page 79 for the formal semantics of $\mathsf{P} \rightsquigarrow^{\mathsf{R},\mathsf{T}}_{(\Delta).\langle \mathsf{s}\rangle} \mathsf{Q}$ and $\mathsf{P} \rightsquigarrow^{\mathsf{R},\mathsf{T}} \mathsf{Q}$.

There are several things worth noting about these assertions; first, they explicitly do *not* enforce stability of the pre- and post-condition. In fact, they allow case analysis on disjunctions and existentials in both P and Q – even inside region assertions. These predicates also satisfy a non-standard rule of consequence that allows weakening of *both* the pre- and post-condition. From this non-standard rule of consequence, we can further derive an asymmetric frame rule that allows arbitrary changes to the context.

Since most of the proof rules for these two predicates are the same, we write $\mathsf{P} \rightsquigarrow^{\mathsf{R},\mathsf{T}}_{\mathcal{J}} \mathsf{Q}$ where $\mathcal{J}$ is defined as follows,

$$\mathcal{J} ::= (\Delta).\langle \mathsf{s}\rangle \mid$$

to refer to both variants at the same time.

$$\frac{\Gamma \vdash \mathsf{R} : \mathsf{Region} \qquad \Gamma \vdash \mathsf{T}_1, \mathsf{T}_2 : \mathsf{RType} \\ \Gamma \mid \Phi \vdash \mathsf{P} \rightsquigarrow^{\mathsf{R},\mathsf{T}_1}_{\mathcal{J}} \mathsf{Q} \qquad \Gamma \mid \Phi \vdash \mathsf{T}_1 \leq \mathsf{T}_2}{\Gamma \mid \Phi \vdash \mathsf{P} \rightsquigarrow^{\mathsf{R},\mathsf{T}_2}_{\mathcal{J}} \mathsf{Q}}$$

$$\frac{}{\Gamma \mid \Phi \vdash \bot \rightsquigarrow^{\mathsf{R},\mathsf{T}}_{\mathcal{J}} \mathsf{Q}} \qquad\qquad \frac{}{\Gamma \mid \Phi \vdash \mathsf{P} \rightsquigarrow^{\mathsf{R},\mathsf{T}}_{\mathcal{J}} \bot}$$

$$\frac{\Gamma \mid \Phi \vdash \mathsf{P}_1 \rightsquigarrow^{\mathsf{R},\mathsf{T}}_{\mathcal{J}} \mathsf{Q} \\ \Gamma \mid \Phi \vdash \mathsf{P}_2 \rightsquigarrow^{\mathsf{R},\mathsf{T}}_{\mathcal{J}} \mathsf{Q}}{\Gamma \mid \Phi \vdash \mathsf{P}_1 \vee \mathsf{P}_2 \rightsquigarrow^{\mathsf{R},\mathsf{T}}_{\mathcal{J}} \mathsf{Q}} \qquad\qquad \frac{\Gamma \mid \Phi \vdash \mathsf{P} \rightsquigarrow^{\mathsf{R},\mathsf{T}}_{\mathcal{J}} \mathsf{Q}_1 \\ \Gamma \mid \Phi \vdash \mathsf{P} \rightsquigarrow^{\mathsf{R},\mathsf{T}}_{\mathcal{J}} \mathsf{Q}_2}{\Gamma \mid \Phi \vdash \mathsf{P} \rightsquigarrow^{\mathsf{R},\mathsf{T}}_{\mathcal{J}} \mathsf{Q}_1 \vee \mathsf{Q}_2}$$

$$\frac{\Gamma \mid \Phi \vdash \boxed{\mathsf{P}_1}^{\mathsf{R}_2,\mathsf{T}_2,\mathsf{A}} * \mathsf{R} \rightsquigarrow^{\mathsf{R}_1,\mathsf{T}_1}_{\mathcal{J}} \mathsf{Q} \\ \Gamma \mid \Phi \vdash \boxed{\mathsf{P}_2}^{\mathsf{R}_2,\mathsf{T}_2,\mathsf{A}} * \mathsf{R} \rightsquigarrow^{\mathsf{R}_1,\mathsf{T}_1}_{\mathcal{J}} \mathsf{Q}}{\Gamma \mid \Phi \vdash \boxed{\mathsf{P}_1 \vee \mathsf{P}_2}^{\mathsf{R}_2,\mathsf{T}_2,\mathsf{A}} * \mathsf{R} \rightsquigarrow^{\mathsf{R}_1,\mathsf{T}_1}_{\mathcal{J}} \mathsf{Q}} \qquad \frac{\Gamma \mid \Phi \vdash \mathsf{P} \rightsquigarrow^{\mathsf{R}_2,\mathsf{A}}_{\mathcal{J}} \boxed{\mathsf{Q}_1}^{\mathsf{R}_1,\mathsf{T}_1} * \mathsf{R} \\ \Gamma \mid \Phi \vdash \mathsf{P} \rightsquigarrow^{\mathsf{R}_2,\mathsf{A}}_{\mathcal{J}} \boxed{\mathsf{Q}_2}^{\mathsf{R}_1,\mathsf{T}_1} * \mathsf{R}}{\Gamma \mid \Phi \vdash \mathsf{P} \rightsquigarrow^{\mathsf{R}_2,\mathsf{A}}_{\mathcal{J}} \boxed{\mathsf{Q}_1 \vee \mathsf{Q}_2}^{\mathsf{R}_1,\mathsf{T}_1} * \mathsf{R}}$$

$$\frac{\Gamma \vdash \mathsf{P}, \mathsf{P}', \mathsf{Q}, \mathsf{Q}' : \mathsf{Prop} \qquad \Gamma \vdash \mathsf{R} : \mathsf{Region} \qquad \Gamma \vdash \mathsf{T} : \mathsf{RType} \\ \Gamma; - \mid \Phi \mid \mathsf{P}' \vdash \mathsf{P} \qquad \Gamma \mid \Phi \vdash \mathsf{P} \rightsquigarrow^{\mathsf{R},\mathsf{T}} \mathsf{Q} \qquad \Gamma; - \mid \Phi \mid \mathsf{Q}' \vdash \mathsf{Q}}{\Gamma \mid \Phi \vdash \mathsf{P}' \rightsquigarrow^{\mathsf{R},\mathsf{T}} \mathsf{Q}'}$$

$$\frac{\begin{array}{c}\Gamma, \Delta \vdash \mathsf{P}, \mathsf{P}', \mathsf{Q}, \mathsf{Q}' : \mathsf{Prop} \qquad \Gamma, \Delta \vdash \mathsf{R} : \mathsf{Region} \qquad \Gamma, \Delta \vdash \mathsf{T} : \mathsf{RType} \\ \Gamma, \Delta \mid \Phi \mid \mathsf{P}' \vdash \mathsf{P} \qquad \Gamma \mid \Phi \vdash \mathsf{P} \rightsquigarrow^{\mathsf{R},\mathsf{T}}_{(\Delta).\langle s \rangle} \mathsf{Q} \qquad \Gamma, \Delta \mid \Phi \mid \mathsf{Q}' \vdash \mathsf{Q}\end{array}}{\Gamma \mid \Phi \vdash \mathsf{P}' \rightsquigarrow^{\mathsf{R},\mathsf{T}}_{(\Delta).\langle s \rangle} \mathsf{Q}'}$$

Any view-shift and atomic update on a region R corresponding to an instance of an action of the protocol on R is permitted, given partial ownership of said action.

$$\frac{\begin{array}{c}\Gamma, \Delta \vdash \mathsf{I}_p, \mathsf{I}_q : \mathsf{Val} \times \mathsf{Action} \times \mathsf{Val} \to \mathsf{Prop} \qquad \Gamma, \Delta \vdash \alpha : \mathsf{Action} \\ \Gamma, \Delta \vdash \mathsf{A}, \mathsf{M} : \mathsf{Val} \\ \Gamma, \Delta \vdash \mathsf{P} : \mathsf{Perm} \qquad \Gamma, \Delta \vdash \mathsf{R} : \mathsf{Region} \qquad \Gamma, \Delta \vdash \mathsf{T}_1, \mathsf{T}_2 : \mathsf{RType} \\ \Gamma, \Delta \mid \Phi \vdash \mathsf{indep}_{\mathsf{T}_2}(\mathsf{I}_p(\mathsf{A}, \alpha, \mathsf{M}), \mathsf{I}_q(\mathsf{A}, \alpha, \mathsf{M})) \qquad \Gamma, \Delta \mid \Phi \vdash \mathsf{T}_2 \not\leq \mathsf{T}_1\end{array}}{\Gamma \mid \Phi \vdash \boxed{\mathsf{I}_p(\mathsf{A}, \alpha, \mathsf{M})}^{\mathsf{R},\mathsf{T}_1,\mathsf{A}}_{\mathsf{I}_p,\mathsf{I}_q} * [\alpha]^{\mathsf{R}}_{\mathsf{P}} \rightsquigarrow^{\mathsf{R},\mathsf{T}_2}_{(\Delta).\langle s \rangle} \boxed{\mathsf{I}_q(\mathsf{A}, \alpha, \mathsf{M})}^{\mathsf{R},\mathsf{T}_1,\mathsf{A}}_{\mathsf{I}_p,\mathsf{I}_q} * [\alpha]^{\mathsf{R}}_{\mathsf{P}}} \text{ AUACT}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathsf{I}_p, \mathsf{I}_q : \mathsf{Val} \times \mathsf{Action} \times \mathsf{Val} \to \mathsf{Prop} \qquad \Gamma \vdash \alpha : \mathsf{Action} \\ \Gamma \vdash \mathsf{A}, \mathsf{M} : \mathsf{Val} \qquad \Gamma \vdash \mathsf{P} : \mathsf{Perm} \qquad \Gamma \vdash \mathsf{R} : \mathsf{Region} \qquad \Gamma \vdash \mathsf{T}_1, \mathsf{T}_2 : \mathsf{RType} \\ \Gamma \mid \Phi \vdash \mathsf{indep}_{\mathsf{T}_2}(\mathsf{I}_p(\mathsf{A}, \alpha, \mathsf{M}), \mathsf{I}_q(\mathsf{A}, \alpha, \mathsf{M})) \qquad \Gamma \mid \Phi \vdash \mathsf{T}_2 \not\leq \mathsf{T}_1\end{array}}{\Gamma \mid \Phi \vdash \boxed{\mathsf{I}_p(\mathsf{A}, \alpha, \mathsf{M})}^{\mathsf{R},\mathsf{T}_1,\mathsf{A}}_{\mathsf{I}_p,\mathsf{I}_q} * [\alpha]^{\mathsf{R}}_{\mathsf{P}} \rightsquigarrow^{\mathsf{R},\mathsf{T}_2} \boxed{\mathsf{I}_q(\mathsf{A}, \alpha, \mathsf{M})}^{\mathsf{R},\mathsf{T}_1,\mathsf{A}}_{\mathsf{I}_p,\mathsf{I}_q} * [\alpha]^{\mathsf{R}}_{\mathsf{P}}} \text{ VSACT}$$

Since $\mathsf{P} \rightsquigarrow^{\mathsf{R},\mathsf{T}}_{(\Delta).\langle \mathsf{s} \rangle} \mathsf{Q}$ assert that any step at the instrumented level from P to Q that corresponds to the atomic statement s at the concrete level is permitted by region R, we can also prove $\mathsf{P} \rightsquigarrow^{\mathsf{R},\mathsf{T}}_{(\Delta).\langle \mathsf{s} \rangle} \mathsf{Q}$ by proving that there exists no such step at the instrumented level. This is useful after performing case analysis to exclude impossible cases, as illustrated by the spin-lock example in Section 3.2. The following rule provides one way of proving that no such step exists at the instrumented level, by providing a concrete field (x.f) whose actual value after the update (M_2) is distinct from the assumed value (M_1).

$$\frac{\begin{array}{c}\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}_1 * \mathsf{P}_2\}\langle s \rangle\{\mathsf{Q}_3\} \qquad \Gamma, \Delta \mid \Phi \vdash \mathsf{M}_1 \neq \mathsf{M}_2 \\ \Gamma, \Delta \mid \Phi \mid \mathsf{Q}_1 * \mathsf{Q}_2 \vdash \mathsf{x.f} \mapsto \mathsf{M}_1 \qquad \Gamma, \Delta \mid \Phi \mid \mathsf{Q}_3 \vdash \mathsf{x.f} \mapsto \mathsf{M}_2\end{array}}{\Gamma \mid \Phi \vdash \boxed{\mathsf{P}_1}^{\mathsf{R},\mathsf{T}_1} * \mathsf{P}_2 \rightsquigarrow^{\mathsf{S},\mathsf{T}_2}_{(\Delta).\langle \mathsf{s} \rangle} \boxed{\mathsf{Q}_1}^{\mathsf{R},\mathsf{T}_1} * \mathsf{Q}_2} \text{ AUFALSE1}$$

Dependence

The dep_T assertion internalizes a notion of region support, allowing explicit reasoning about the absence of self-referential region and protocol assertions. In particular, $\mathsf{dep}_\mathsf{T}(\mathsf{P})$ asserts that P is invariant under arbitrary changes to shared regions and protocols of regions with region types not greater than or equal to T. Conversely, $\mathsf{indep}_\mathsf{T}(\mathsf{P})$ asserts that P is invariant under arbitrary changes to shared regions and protocols of regions with region types greater than or equal to T.

$$\frac{\Gamma \vdash \mathsf{T} : \mathsf{RType}}{\Gamma \mid \Phi \vdash \mathsf{dep}_{\mathsf{T}}(\bot)} \qquad \frac{\Gamma \vdash \mathsf{T} : \mathsf{RType}}{\Gamma \mid \Phi \vdash \mathsf{dep}_{\mathsf{T}}(\top)} \qquad \frac{\begin{array}{c}\Gamma \vdash \mathsf{S} : \mathsf{Spec} \\ \Gamma \vdash \mathsf{T} : \mathsf{RType}\end{array}}{\Gamma \mid \Phi \vdash \mathsf{dep}_{\mathsf{T}}(\mathsf{asn}(\mathsf{S}))}$$

$$\frac{\Gamma \mid \Phi \vdash \mathsf{dep}_{\mathsf{T}}(\mathsf{P}) \qquad \Gamma \mid \Phi \vdash \mathsf{dep}_{\mathsf{T}}(\mathsf{Q}) \qquad op \in \{\wedge, \vee, *, \Rightarrow\}}{\Gamma \mid \Phi \vdash \mathsf{dep}_{\mathsf{T}}(\mathsf{P}\ op\ \mathsf{Q})}$$

$$\frac{\Gamma \mid \Phi \vdash \forall x : \tau.\ \mathsf{dep}_{\mathsf{T}}(\mathsf{P}) \qquad Q \in \{\forall, \exists\}}{\Gamma \mid \Phi \vdash \mathsf{dep}_{\mathsf{T}}(Qx : \tau.\ \mathsf{P})} \qquad \frac{\begin{array}{c}\Gamma \mid \Phi \vdash \mathsf{T}_1 \leq \mathsf{T}_2 \\ \Gamma \mid \Phi \vdash \mathsf{dep}_{\mathsf{T}_2}(\mathsf{P})\end{array}}{\Gamma \mid \Phi \vdash \mathsf{dep}_{\mathsf{T}_1}(\mathsf{P})}$$

$$\frac{\Gamma \mid \Phi \vdash \mathsf{dep}_{\mathsf{T}_1}(\mathsf{P})}{\Gamma \mid \Phi \vdash \mathsf{dep}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}(\boxed{\mathsf{P}}^{\mathsf{R},\mathsf{T}_2,\mathsf{A}})} \qquad \frac{\begin{array}{c}\Gamma \mid \Phi \vdash \forall x : \mathsf{Val} \times \mathsf{Action} \times \mathsf{Val}.\ \mathsf{dep}_{\mathsf{T}_1}(\mathsf{I}_p(x)) \\ \Gamma \mid \Phi \vdash \forall x : \mathsf{Val} \times \mathsf{Action} \times \mathsf{Val}.\ \mathsf{dep}_{\mathsf{T}_1}(\mathsf{I}_q(x))\end{array}}{\Gamma \mid \Phi \vdash \mathsf{dep}_{\mathsf{T}_1 \sqcap \mathsf{T}_2}(\mathsf{protocol}(\mathsf{T}_2, \mathsf{I}_p, \mathsf{I}_q))}$$

$$\frac{\Gamma \mid \Phi \vdash \mathsf{dep}_{\mathsf{T}_1}(\mathsf{P}) \qquad \Gamma \mid \Phi \vdash \mathsf{T}_1 \not\leq \mathsf{T}_2 \qquad \Gamma \mid \Phi \vdash \mathsf{T}_2 \not\leq \mathsf{T}_1}{\Gamma \mid \Phi \vdash \mathsf{indep}_{\mathsf{T}_2}(\mathsf{P})}$$

$$\frac{\Gamma \mid \Phi \vdash \mathsf{T}_1 \not\leq \mathsf{T}_2 \qquad \Gamma \mid \Phi \vdash \mathsf{indep}_{\mathsf{T}_1}(\mathsf{P})}{\Gamma \mid \Phi \vdash \mathsf{indep}_{\mathsf{T}_1}(\boxed{\mathsf{P}}^{\mathsf{R},\mathsf{T}_2,\mathsf{A}})}$$

Protocol purity

The $\mathsf{pure}_{\mathsf{protocol}}$ predicate expresses that a given assertion is invariant under arbitrary changes to currently allocated protocols. The $\mathsf{pure}_{\mathsf{protocol}}$ predicate is closed under all the usual connectives.

$$\frac{}{\Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{protocol}}(\bot)} \qquad \frac{}{\Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{protocol}}(\top)}$$

$$\frac{\Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{protocol}}(\mathsf{P}) \qquad \Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{protocol}}(\mathsf{Q}) \qquad op \in \{\wedge, \vee, *, \Rightarrow\}}{\Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{protocol}}(\mathsf{P}\ op\ \mathsf{Q})}$$

$$\frac{\Gamma \mid \Phi \vdash \forall \mathsf{x} : \tau.\ \mathsf{pure}_{\mathsf{protocol}}(\mathsf{P}) \qquad Q \in \{\exists, \forall\}}{\Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{protocol}}(Q\mathsf{x} : \tau.\ \mathsf{P})}$$

The $\mathsf{pure}_{\mathsf{protocol}}$ predicate is also closed under the mini C$^\sharp$ specific state assertions.

$$\frac{\begin{array}{c}\Gamma \vdash \mathsf{M} : \mathsf{Val} \\ \Gamma \vdash \mathsf{F} : \mathsf{Field} \qquad \Gamma \vdash \mathsf{N} : \mathsf{Val}\end{array}}{\Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{protocol}}(\mathsf{M.F} \mapsto \mathsf{N})} \qquad \frac{\begin{array}{c}\Gamma \vdash \mathsf{M} : \mathsf{Val} \\ \Gamma \vdash \mathsf{F} : \mathsf{Field} \qquad \Gamma \vdash \mathsf{N} : \mathsf{Val}\end{array}}{\Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{protocol}}(\mathsf{M}_{\mathsf{F}} \mapsto \mathsf{N})}$$

$$\frac{\Gamma \vdash \mathsf{N}_1, \mathsf{N}_2 : \mathsf{Val} \qquad \Gamma \vdash \mathsf{M} : \mathsf{Method}}{\Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{protocol}}(\mathsf{N}_1 \mapsto \mathsf{N}_2.\mathsf{M})} \qquad \frac{\Gamma \vdash \mathsf{M} : \mathsf{Val} \qquad \Gamma \vdash \mathsf{C} : \mathsf{Class}}{\Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{protocol}}(\mathsf{M} : \mathsf{C})}$$

It is also closed under region and action assertions, but not protocol assertions.

$$\frac{\begin{array}{c}\Gamma \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma \vdash \mathsf{R} : \mathsf{Region} \qquad \Gamma \vdash \mathsf{T} : \mathsf{RType} \qquad \Gamma \vdash \mathsf{A} : \mathsf{Val} \\ \Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{protocol}}(\mathsf{P})\end{array}}{\Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{protocol}}\left(\boxed{\mathsf{P}}^{\mathsf{R,T,A}}\right)}$$

$$\frac{\Gamma \vdash \mathsf{A} : \mathsf{Action} \qquad \Gamma \vdash \mathsf{R} : \mathsf{Region} \qquad \Gamma \vdash \mathsf{P} : \mathsf{Perm}}{\Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{protocol}}([\mathsf{A}]^{\mathsf{R}}_{\mathsf{P}})}$$

Lastly, $\mathsf{pure}_{\mathsf{protocol}}$ is closed under arbitrary embedded specifications.

$$\frac{\Gamma \vdash \mathsf{S} : \mathsf{Spec}}{\Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{protocol}}(\mathsf{asn}(\mathsf{S}))}$$

State purity

The $\mathsf{pure}_{\mathsf{state}}$ predicate expresses that a given assertion is invariant under arbitrary changes to the current local and shared state. The $\mathsf{pure}_{\mathsf{state}}$ predicate is closed under all the usual connectives.

$$\frac{}{\Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{state}}(\bot)} \qquad \frac{}{\Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{state}}(\top)}$$

$$\frac{\Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{state}}(\mathsf{P}) \qquad \Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{state}}(\mathsf{Q}) \qquad op \in \{\wedge, \vee, *, \Rightarrow\}}{\Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{state}}(\mathsf{P}\ op\ \mathsf{Q})}$$

$$\frac{\Gamma \mid \Phi \vdash \forall \mathsf{x} : \tau.\ \mathsf{pure}_{\mathsf{state}}(\mathsf{P}) \qquad Q \in \{\exists, \forall\}}{\Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{state}}(Q\mathsf{x} : \tau.\ \mathsf{P})}$$

It is also closed under protocol assertions, but not region or action assertions.

$$\frac{\Gamma \vdash \mathsf{T} : \mathsf{RType} \qquad \Gamma \vdash \mathsf{I}_p, \mathsf{I}_q : \mathsf{Val} \times \mathsf{Action} \times \mathsf{Val} \to \mathsf{Prop}}{\Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{state}}(\mathsf{protocol}(\mathsf{T}, \mathsf{I}_p, \mathsf{I}_q))}$$

The $\mathsf{pure}_{\mathsf{state}}$ predicate is also closed under embeddings of arbitrary specifications.

$$\frac{\Gamma \vdash \mathsf{S} : \mathsf{Spec}}{\Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{state}}(\mathsf{asn}(\mathsf{S}))}$$

Permission purity

The $\mathsf{pure_{perm}}$ predicate asserts that a given assertion is invariant under arbitrary ownership transfer of action permissions. It is closed under all the usual connectives.

$$\frac{}{\Gamma \mid \Phi \vdash \mathsf{pure_{perm}}(\bot)} \qquad \frac{}{\Gamma \mid \Phi \vdash \mathsf{pure_{perm}}(\top)}$$

$$\frac{\Gamma \mid \Phi \vdash \mathsf{pure_{perm}}(\mathsf{P}) \qquad \Gamma \mid \Phi \vdash \mathsf{pure_{perm}}(\mathsf{Q}) \qquad op \in \{\wedge, \vee, *, \Rightarrow\}}{\Gamma \mid \Phi \vdash \mathsf{pure_{perm}}(\mathsf{P}\ op\ \mathsf{Q})}$$

$$\frac{\Gamma \mid \Phi \vdash \forall \mathsf{x} : \tau.\ \mathsf{pure_{perm}}(\mathsf{P}) \qquad Q \in \{\exists, \forall\}}{\Gamma \mid \Phi \vdash \mathsf{pure_{perm}}(Q\mathsf{x} : \tau.\ \mathsf{P})}$$

The $\mathsf{pure_{perm}}$ predicate is also closed under the mini C$^\sharp$ specific state assertions.

$$\frac{\begin{array}{c}\Gamma \vdash \mathsf{M} : \mathsf{Val} \\ \Gamma \vdash \mathsf{F} : \mathsf{Field} \qquad \Gamma \vdash \mathsf{N} : \mathsf{Val}\end{array}}{\Gamma \mid \Phi \vdash \mathsf{pure_{perm}}(\mathsf{M.F} \mapsto \mathsf{N})} \qquad \frac{\begin{array}{c}\Gamma \vdash \mathsf{M} : \mathsf{Val} \\ \Gamma \vdash \mathsf{F} : \mathsf{Field} \qquad \Gamma \vdash \mathsf{N} : \mathsf{Val}\end{array}}{\Gamma \mid \Phi \vdash \mathsf{pure_{perm}}(\mathsf{M_F} \mapsto \mathsf{N})}$$

$$\frac{\Gamma \vdash \mathsf{N_1}, \mathsf{N_2} : \mathsf{Val} \qquad \Gamma \vdash \mathsf{M} : \mathsf{Method}}{\Gamma \mid \Phi \vdash \mathsf{pure_{perm}}(\mathsf{N_1} \mapsto \mathsf{N_2.M})} \qquad \frac{\Gamma \vdash \mathsf{M} : \mathsf{Val} \qquad \Gamma \vdash \mathsf{C} : \mathsf{Class}}{\Gamma \mid \Phi \vdash \mathsf{pure_{perm}}(\mathsf{M} : \mathsf{C})}$$

It is also closed under region and protocol assertions, but not action assertions.

$$\frac{\begin{array}{c}\Gamma \vdash \mathsf{P} : \mathsf{Prop} \qquad \Gamma \vdash \mathsf{R} : \mathsf{Region} \qquad \Gamma \vdash \mathsf{T} : \mathsf{RType} \qquad \Gamma \vdash \mathsf{A} : \mathsf{Val} \\ \Gamma \mid \Phi \vdash \mathsf{pure_{perm}}(\mathsf{P})\end{array}}{\Gamma \mid \Phi \vdash \mathsf{pure_{perm}}\left(\boxed{\mathsf{P}}^{\mathsf{R,T,A}}\right)}$$

$$\frac{\Gamma \vdash \mathsf{T} : \mathsf{RType} \qquad \Gamma \vdash \mathsf{I}_p, \mathsf{I}_q : \mathsf{Val} \times \mathsf{Action} \times \mathsf{Val} \to \mathsf{Prop}}{\Gamma \mid \Phi \vdash \mathsf{pure_{perm}}(\mathsf{protocol}(\mathsf{T}, \mathsf{I}_p, \mathsf{I}_q))}$$

Lastly, $\mathsf{pure_{perm}}$ is closed under embeddings of arbitrary specifications.

$$\frac{\Gamma \vdash \mathsf{S} : \mathsf{Spec}}{\Gamma \mid \Phi \vdash \mathsf{pure_{perm}}(\mathsf{asn}(\mathsf{S}))}$$

Phantom state

Hoare logics commonly feature auxiliary variables to record an abstraction of the state and history of execution. Auxiliary variables are updated through standard assignments, but they are not allowed to affect the flow of execution. Phantom fields provide a purely logical notion of an auxiliary variable. Phantom fields are updated through view-shifts in the logic, instead of assignments in code. Updating a phantom fields requires full ownership of said field.

$$\frac{\Gamma \vdash \mathsf{M} : \mathsf{Val} \qquad \Gamma \vdash \mathsf{F} : \mathsf{Field} \qquad \Gamma \vdash \mathsf{N} : \mathsf{Val}}{\Gamma \mid \Phi \vdash \mathsf{M}_\mathsf{F} \mapsto \_ \sqsubseteq \mathsf{M}_\mathsf{F} \mapsto \mathsf{N}}$$

Like ordinary fields, phantom fields have a fixed value at any given point in time.

$$\frac{\Gamma \vdash \mathsf{M} : \mathsf{Val} \qquad \Gamma \vdash \mathsf{F} : \mathsf{Field} \qquad \Gamma \vdash \mathsf{N}_1, \mathsf{N}_2 : \mathsf{Val}}{\Gamma \mid \Phi \vdash \mathsf{M}_\mathsf{F} \mapsto \mathsf{N}_1 \wedge \mathsf{M}_\mathsf{F} \mapsto \mathsf{N}_2 \Rightarrow \mathsf{N}_1 =_{\mathsf{Val}} \mathsf{N}_2}$$

Phantom fields are allocated in the proof rule for constructor verification.

2.3.5 Hoare logic

Structural rules

$$\frac{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}\}\mathsf{s}\{\mathsf{Q}\} \qquad \Gamma, \Delta \mid \Phi \vdash \mathsf{stable}(\mathsf{R}) \qquad \mathsf{mod}(\mathsf{s}) \cap \mathsf{FV}(\mathsf{R}) = \emptyset}{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P} * \mathsf{R}\}\mathsf{s}\{\mathsf{Q} * \mathsf{R}\}}$$

$$\frac{\begin{array}{c}\Gamma, \Delta \mid \Phi \vdash \mathsf{stable}(\mathsf{P}) \wedge \mathsf{stable}(\mathsf{Q}) \\ \Gamma, \Delta \mid \Phi \vdash \mathsf{P} \sqsubseteq \mathsf{P}' \qquad \Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}'\}\mathsf{s}\{\mathsf{Q}'\} \qquad \Gamma, \Delta \mid \Phi \vdash \mathsf{Q}' \sqsubseteq \mathsf{Q}\end{array}}{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}\}\mathsf{s}\{\mathsf{Q}\}}$$

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

Primitive statements

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

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

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

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

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

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

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

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

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

Method verification

To verify a method, we verify the method body.

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

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 (See definition of erasure on page 62 for the formal semantics). The constructor verification rule thus allows the user to introduce an arbitrary (finite) number of phantom fields on the newly created object.

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

3 Examples

In this section we present two examples to illustrate how to use the proof system. The first example illustrates reasoning about recursion through the store using guarded recursion. The second example illustrates higher-order concurrent abstract predicates using a spin-lock.

3.1 Recursion through the store

To illustrate how the embedding of specifications in assertions combined with guarded recursion allows us to reason about recursion through the store, consider the following program.

```
using System;

class NatRec {
  private Func<NatRec, int, int> r = null;

  public NatRec(Func<NatRec, int, int> r) {
    this.r = r;
  }

  public int comp(int n) {
    return r(this, n);
  }
}

class Factorial {
  private int f(NatRec r, int n) {
    if (n == 0) return 1; else return n * r.comp (n-1);
  }

  public static void fac(int m) {
    NatRec fac = new NatRec(f);
    return fac.comp(m);
  }
}
```

This program implements a factorial computation using recursion through the store. In particular, the NatRec constructor takes as argument a delegate that itself takes as argument a NatRec instance and an integer, and returns an integer. When this delegate is invoked in the comp method, NatRec calls the delegate with a reference to itself, allowing the delegate to recursively call itself through the comp method. Since the factorial client explicitly exploits this behavior of the NatRec class, we thus have to give a NatRec specification that explicitly allows the client delegate to call the comp method on its NatRec argument, and that this recursive call returns the factorial of its argument.

To specify NatRec, we require the client to choose a function on natural numbers ($f : \mathbb{N} \to \mathbb{N}$), representing the intended computation. The NatRec constructor then requires the client to provide a delegate that implements f, assuming that calling comp on its NatRec argument, implements f. In this case, comp implements f. We can thus specify NatRec as follows.

$$\begin{aligned} S_{\mathsf{NatRec}} \stackrel{\text{def}}{=} \forall \mathsf{f} \in \mathbb{N} \to \mathbb{N}.\ \exists \mathsf{rec} : \mathsf{Val} \to \mathsf{Prop}.& \\ \mathtt{NatRec} : (\mathsf{r}).\{I_{del}(\mathsf{rec}, \mathsf{f}, \mathsf{r})\}\{\mathsf{rec}(\mathtt{this})\}\ \wedge& \\ \mathtt{NatRec.comp} : (\mathsf{n}).\{\mathsf{rec}(\mathtt{this})\}\{\mathsf{ret}.\ \mathsf{rec}(\mathtt{this}) \wedge \mathsf{ret} = \mathsf{f}(\mathsf{n})\}& \end{aligned}$$

where $I_{del} : (\mathsf{Val} \to \mathsf{Prop}) \times (\mathbb{N} \to \mathbb{N}) \times \mathsf{Val} \to \mathsf{Prop}$ is defined as

$$I_{del}(\mathsf{rec}, \mathsf{f}, \mathsf{y}) \stackrel{\text{def}}{=} \mathsf{y} \mapsto (\mathsf{x}, \mathsf{n}).\{\mathsf{rec}(\mathsf{x})\}\{\mathsf{ret}.\ \mathsf{rec}(\mathsf{x}) * \mathsf{ret} = \mathsf{f}(\mathsf{n})\}$$

Lemma 1.

$$- \mid - \vdash S_{NatRec}$$

Proof.

- suppose $\mathsf{f} : \mathbb{N} \to \mathbb{N}$
- define $\mathsf{rec}' : (\mathsf{Val} \to \mathsf{Prop}) \to (\mathsf{Val} \to \mathsf{Prop})$ as the following functional

$$\mathsf{rec}'(\mathsf{p})(\mathsf{x}) \stackrel{\text{def}}{=} \exists \mathsf{y} : \mathsf{Val}.\ \mathsf{x.r} \mapsto \mathsf{y} * \triangleright I_{del}(\mathsf{p}, \mathsf{f}, \mathsf{y})$$

- define rec as the fixed-point of rec′

$$\mathsf{rec} \stackrel{\text{def}}{=} \mathsf{fix}(\mathsf{rec}')$$

- since p is guarded in rec′, we have that

$$\mathsf{rec}(\mathsf{x}) \Leftrightarrow \exists \mathsf{y} : \mathsf{Val}.\ \mathsf{x.r} \mapsto \mathsf{y} * \triangleright I_{del}(\mathsf{p}, \mathsf{f}, \mathsf{y})$$

- to prove the constructor we thus need to "forget a step"

```
public NatRec(Func<NatRec, int, int> r) {
      {this.r ↦ null ∗ $I_{del}$(rec, f, r)}
      {this.r ↦ null ∗ ▷$I_{del}$(rec, f, r)}
   this.r = r;
      {this.r ↦ r ∗ ▷$I_{del}$(rec, f, r)}
      {rec(this)}
}
```

- and the proof of comp is similarly straight-forward

```
public int comp(int n) {
      {rec(this)}
   Func<NatRec, int, int> y; int z;
      {rec(this)}
      {∃y : Val. this.r ↦ y ∗ ▷$I_{del}$(rec, f, y)}
```

```
    y = this.r;
      {this.r ↦ y ∗ ▷I_del(rec, f, y)}
      {this.r ↦ y ∗ ▷I_del(rec, f, y) ∗ ▷I_del(rec, f, y)}
      {rec(this) ∗ ▷I_del(rec, f, y)}
      {rec(this) ∗ ▷f ↦ (x, n).{rec(x)}{ret.  rec(x) ∗ ret =_ℕ f(n)}}
    z = f(this, n);
      {rec(this) ∗ z =_ℕ f(n)}
    return z;
      {rec(this) ∗ ret =_ℕ f(n)}
}
```

□

Lemma 2.

$$- \mid S_{NatRec} \vdash \mathit{Factorial.fac} : (\mathsf{n}).\{\mathsf{emp}\}\{\mathsf{ret} = \mathsf{n}!\}$$

Proof.

- Pick f to be $! : \mathbb{N} \to \mathbb{N}$ in S_{NatRec}
- Then we first need to show that

  $$\mathsf{Factorial.f} : (\mathsf{x}, \mathsf{n}).\{\mathsf{rec}(\mathsf{x})\}\{\mathsf{rec}(\mathsf{x}) * \mathsf{ret} = \mathsf{n}!\}$$

  which follows by a straight-foward proof:

```
private int f(NatRec x, int n) {
    int m;
      {rec(x)}
    if (n == 0)
        {rec(x) ∗ n = 0}
      m = 1;
        {rec(x) ∗ m = n!}
    else {
        {rec(x)}
      m = x.comp(n-1);
        {rec(x) ∗ m = (n − 1)!}
      m = n * m;
        {rec(x) ∗ m = n!}
    }
      {rec(x) ∗ m = n!}
    return m;
      {ret.  rec(x) ∗ ret = n!}
}
```

- Now, we can embed this specification for f to derive the desired fac specification:

```
public static void fac(int n) {
  Func<NatRec, int, int> g; NatRec fac; int m;
    {emp}
  g = this.f;
    {g ↦ (x, n).{rec(x)}{rec(x) * ret = n!}
  fac = new NatRec(g);
    {rec(fac)}
  m = fac.comp(n);
    {rec(fac) * m = n!}
  return m;
    {ret. ret = n!}
}
```

□

3.2 Spin lock

To illustrate the use of higher-order concurrent abstract predicates, consider the following spin-lock.

```
class Lock {
  private int locked = 0;

  public void Acquire() {
    x = CAS(this.locked, 0, 1);
    if(x != null)
      Acquire();
  }

  public void Release() {
    locked = 0;
  }
}
```

It uses the private field locked to maintain the current state of the lock (0 for unlocked, 1 for locked) and a compare-and-swap to atomically acquire the lock.

The standard separation logic specification of a lock, associates a resource invariant R with each lock, which describes the resources protected by the lock. The resource invariant is thus transferred to the client upon acquiring the lock, and transferred back upon releasing the lock. Since the resource invariant R might itself assert ownership of shared resources using CAP, we require that R is stable, that it is independent of the lock region type (picked by the client), and that it is expressible using state-independent protocols:

$$\begin{aligned}
S_{\mathsf{Lock}} = \ & \exists \mathsf{islock}, \mathsf{locked} : \mathsf{RType} \times \mathsf{Prop} \times \mathsf{Val} \to \mathsf{Prop}. \\
& \quad \forall \mathsf{t} \in \mathsf{RType}.\ \forall \mathsf{R} : \mathsf{Prop}.\ \mathsf{indep_t}(\mathsf{R}) \wedge \mathsf{sip}(\mathsf{R}) \wedge \mathsf{stable}(\mathsf{R}) \ \Rightarrow \\
& \qquad \mathsf{new\ Lock}() : (\text{-}).\ \{\mathsf{R}\}\ \{\mathsf{ret}.\ \exists \mathsf{s} : \mathsf{RType}.\ \mathsf{isLock}(\mathsf{s}, \mathsf{R}, \mathsf{ret}) * \mathsf{t} \leq \mathsf{s}\} \\
& \qquad \mathsf{Lock.Acquire} : (\text{-}).\ \{\mathsf{isLock}(\mathsf{t}, \mathsf{R}, \mathtt{this})\}\ \{\mathsf{locked}(\mathsf{t}, \mathsf{R}, \mathtt{this}) * \mathsf{R}\} \\
& \qquad \mathsf{Lock.Release} : (\text{-}).\ \{\mathsf{locked}(\mathsf{t}, \mathsf{R}, \mathtt{this}) * \mathsf{R}\}\ \{\mathsf{isLock}(\mathsf{t}, \mathsf{R}, \mathtt{this})\} \\
& \qquad \forall \mathsf{x} : \mathsf{Val}.\ \mathsf{valid}(\mathsf{isLock}(\mathsf{t}, \mathsf{R}, \mathsf{x}) \Leftrightarrow \mathsf{isLock}(\mathsf{t}, \mathsf{R}, \mathsf{x}) * \mathsf{isLock}(\mathsf{t}, \mathsf{R}, \mathsf{x})) \ \wedge \\
& \qquad\qquad \mathsf{stable}(\mathsf{isLock}(\mathsf{t}, \mathsf{R}, \mathsf{x})) \wedge \mathsf{stable}(\mathsf{locked}(\mathsf{t}, \mathsf{R}, \mathsf{x}))
\end{aligned}$$

The $\mathsf{isLock}(\mathsf{t}, \mathsf{R}, \mathsf{x})$ predicate asserts that x refers to a lock with resource invariant R. The $\mathsf{isLock}(-)$ predicate is freely duplicable, allowing multiple clients to use the same lock. Likewise, the $\mathsf{locked}(\mathsf{t}, \mathsf{R}, \mathsf{x})$ predicate asserts that x refers to a locked lock with resource invariant R, and the permission to unlock it; $\mathsf{locked}(-)$ is thus not duplicable.

Lemma 3.

$$- \mid - \vdash S_{Lock}$$

Proof.

- Define the representation predicates $\mathsf{isLock}(-)$ and $\mathsf{locked}(-)$ as follows

  $$\begin{aligned}
  & \mathsf{isLock}(\mathsf{t}, \mathsf{R}, \mathsf{x}) \overset{\text{def}}{=} \\
  & \quad \exists \mathsf{r} : \mathsf{Region}.\ \exists \pi : \mathsf{Perm}.\ \mathsf{asn}(\mathsf{stable}(\mathsf{R}) \wedge \mathsf{indep_t}(\mathsf{R}) \wedge \mathsf{sip}(\mathsf{R})) \\
  & \quad * \ \boxed{(\mathsf{x.locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{\mathsf{r}}_{1}) \vee \mathsf{x.locked} \mapsto 1}^{\mathsf{r}, \mathsf{t}, (\mathsf{x}, \mathsf{r})}_{\mathsf{I}(\mathsf{R})} * [\textsc{Lock}]^{\mathsf{r}}_{\pi} * [\tau_1]^{\mathsf{r}}_{\pi} * [\tau_2]^{\mathsf{r}}_{\pi} \\
  & \mathsf{locked}(\mathsf{t}, \mathsf{R}, \mathsf{x}) \overset{\text{def}}{=} \\
  & \quad \exists \mathsf{r} : \mathsf{Region}.\ \exists \pi : \mathsf{Perm}.\ \mathsf{asn}(\mathsf{stable}(\mathsf{R}) \wedge \mathsf{indep_t}(\mathsf{R}) \wedge \mathsf{sip}(\mathsf{R})) \\
  & \quad * \ \boxed{\mathsf{x.locked} \mapsto 1}^{\mathsf{r}, \mathsf{t}, (\mathsf{x}, \mathsf{r})}_{\mathsf{I}(\mathsf{R})} * [\textsc{Lock}]^{\mathsf{r}}_{\pi} * [\tau_1]^{\mathsf{r}}_{\pi} * [\tau_2]^{\mathsf{r}}_{\pi} * [\textsc{Unlock}]^{\mathsf{r}}_{1}
  \end{aligned}$$

  where I is the parametric protocol:

  $$\mathsf{I}(\mathsf{R})(\mathsf{x}, \mathsf{r}) \overset{\text{def}}{=} \left(\begin{array}{r@{\ :\ }l}
  \textsc{Lock} & \mathsf{x.locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{\mathsf{r}}_{1} \rightsquigarrow \mathsf{x.locked} \mapsto 1 \\
  \textsc{Unlock} & \mathsf{x.locked} \mapsto 1 \rightsquigarrow \mathsf{x.locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{\mathsf{r}}_{1} \\
  [\tau_1] & \mathsf{x.locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{\mathsf{r}}_{1} \rightsquigarrow \mathsf{x.locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{\mathsf{r}}_{1} \\
  [\tau_2] & \mathsf{x.locked} \mapsto 1 \rightsquigarrow \mathsf{x.locked} \mapsto 1
  \end{array} \right)$$

- Next, we have to prove the representation predicates stable; we sketch the proof for $\mathsf{isLock}(-)$; the proof for $\mathsf{locked}(-)$ is simpler. To prove $\mathsf{isLock}(-)$ stable, first we apply rule StableAsn to extract the embedded specification assumptions from $\mathsf{isLock}(-)$ about the resource invariant. It thus suffices to prove,

  $$\mathsf{t} : \mathsf{RType}, \mathsf{R} : \mathsf{Prop}, \mathsf{x} : \mathsf{Val} \mid \mathsf{stable}(\mathsf{R}), \mathsf{indep_t}(\mathsf{R}), \mathsf{sip}(\mathsf{R}) \vdash \mathsf{stable}(\mathsf{isLock}(\mathsf{t}, \mathsf{R}, \mathsf{x}))$$

From the $\mathsf{sip}(\mathsf{R})$ assumption, there exists $\mathsf{S}, \mathsf{P} : \mathsf{Prop}$ such that $\mathsf{pure}_{\mathsf{protocol}}(\mathsf{S})$, $\mathsf{pure}_{\mathsf{state}}(\mathsf{P})$ and $\mathsf{valid}(\mathsf{R} \Leftrightarrow (\mathsf{S} * \mathsf{P}))$. We can use this decomposition of R into S and P to pull out the region-assertions of R from the shared region in $\mathsf{isLock}(-)$. That is, in the given context,

$$\boxed{(\mathsf{x.locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{\mathsf{r}}_{1}) \vee \mathsf{x.locked} \mapsto 1}^{\mathsf{r,t,(x,r)}}_{\mathsf{I(R)}} \Leftrightarrow$$
$$\boxed{(\mathsf{x.locked} \mapsto 0 * \mathsf{S} * [\textsc{Unlock}]^{\mathsf{r}}_{1}) \vee \mathsf{x.locked} \mapsto 1}^{\mathsf{r,t,(x,r)}}_{\mathsf{I(R)}} * \mathsf{P}$$

Stability of $\mathsf{isLock}(-)$ thus reduces to,

$$\Gamma \mid \Phi \vdash \mathsf{stable}(\boxed{(\mathsf{x.locked} \mapsto 0 * \mathsf{S} * [\textsc{Unlock}]^{\mathsf{r}}_{1}) \vee \mathsf{x.locked} \mapsto 1}^{\mathsf{r,t,(x,r)}}_{\mathsf{I(R)}} * \mathsf{P})$$

where

$$\Gamma = \mathsf{t} : \mathsf{RType}, \mathsf{R}, \mathsf{S}, \mathsf{P} : \mathsf{Prop}, \mathsf{x} : \mathsf{Val}, \mathsf{r} : \mathsf{Region}, \pi : \mathsf{Perm}$$
$$\Phi = \mathsf{stable}(\mathsf{S} * \mathsf{P}), \mathsf{indep}_{\mathsf{t}}(\mathsf{S} * \mathsf{P}), \mathsf{pure}_{\mathsf{protocol}}(\mathsf{S}), \mathsf{pure}_{\mathsf{state}}(\mathsf{P}), \mathsf{valid}(\mathsf{R} \Leftrightarrow \mathsf{S} * \mathsf{P})$$

By StableD this decomposes into stability under every action. We prove stability under the Lock and Unlock actions, the remaining actions are similar. We thus have to show that,

$$\Gamma \mid \Phi \vdash \mathsf{stable}^{\mathsf{r}}_{\textsc{Lock}}(\boxed{(\mathsf{x.locked} \mapsto 0 * \mathsf{S} * [\textsc{Unlock}]^{\mathsf{r}}_{1}) \vee \mathsf{x.locked} \mapsto 1}^{\mathsf{r,t,(x,r)}}_{\mathsf{I(R)}} * \mathsf{P})$$

and

$$\Gamma \mid \Phi \vdash \mathsf{stable}^{\mathsf{r}}_{\textsc{Unlock}}(\boxed{(\mathsf{x.locked} \mapsto 0 * \mathsf{S} * [\textsc{Unlock}]^{\mathsf{r}}_{1}) \vee \mathsf{x.locked} \mapsto 1}^{\mathsf{r,t,(x,r)}}_{\mathsf{I(R)}} * \mathsf{P})$$

To apply the StableClosed rule we need to prove that,

$$\Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{protocol}}((\mathsf{x.locked} \mapsto 0 * \mathsf{S} * [\textsc{Unlock}]^{\mathsf{r}}_{1}) \vee \mathsf{x.locked} \mapsto 1)$$
$$\Gamma \mid \Phi \vdash \mathsf{pure}_{\mathsf{state}}(\mathsf{P})$$
$$\Gamma \mid \Phi \vdash \mathsf{stable}(((\mathsf{x.locked} \mapsto 0 * \mathsf{S} * [\textsc{Unlock}]^{\mathsf{r}}_{1}) \vee \mathsf{x.locked} \mapsto 1) * \mathsf{P})$$
$$\Gamma \mid \Phi \vdash \mathsf{indep}_{\mathsf{t}}(((\mathsf{x.locked} \mapsto 0 * \mathsf{S} * [\textsc{Unlock}]^{\mathsf{r}}_{1}) \vee \mathsf{x.locked} \mapsto 1) * \mathsf{P})$$

All of these follow fairly easily from the assumptions in Φ. Stability under the Lock action thus reduces to

$$\begin{aligned}\Gamma \mid \Phi \vdash\ & \mathsf{valid}(((\mathsf{x.locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{r}_{1}) \wedge \\ &\qquad ((\mathsf{x.locked} \mapsto 0 * \mathsf{S} * [\textsc{Unlock}]^{\mathsf{r}}_{1}) \vee \mathsf{x.locked} \mapsto 1)) \Rightarrow \bot) \vee \\ & \mathsf{valid}(\mathsf{x.locked} \mapsto 1 \Rightarrow ((\mathsf{x.locked} \mapsto 0 * \mathsf{S} * [\textsc{Unlock}]^{\mathsf{r}}_{1}) \vee \mathsf{x.locked} \mapsto 1))\end{aligned}$$

and stability under the UNLOCK action reduces to

$$\begin{aligned}\Gamma \mid \Phi \vdash \ & \mathsf{valid}(((\mathsf{x.locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{\mathsf{r}}_{1}) \wedge \\ & \qquad\qquad ((\mathsf{x.locked} \mapsto 0 * \mathsf{S} * [\textsc{Unlock}]^{\mathsf{r}}_{1}) \vee \mathsf{x.locked} \mapsto 1)) \Rightarrow \bot) \vee \\ & \mathsf{valid}((\mathsf{x.locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{\mathsf{r}}_{1}) \ \Rightarrow \\ & \qquad\qquad ((\mathsf{x.locked} \mapsto 0 * \mathsf{S} * [\textsc{Unlock}]^{\mathsf{r}}_{1}) \vee \mathsf{x.locked} \mapsto 1))\end{aligned}$$

In both cases we prove the second disjunct. The first obligation is trivial. The second obligation follows from the assumption that $\mathsf{valid}(\mathsf{R} \Leftrightarrow \mathsf{S} * \mathsf{P})$.

- Next, we have to verify the lock methods. Below we give a rough proof-sketch of Lock.Acquire. Formally, we first use the LOEB rule to perform Löb induction, to reason about the recursive call.

```
public void Acquire() {
    ${\{\mathsf{isLock}(\mathsf{t}, \mathsf{R}, \mathtt{this})\}}$
    $\{\boxed{(\mathtt{this}.\mathsf{locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{\mathsf{r}}_{1}) \vee \mathtt{this}.\mathsf{locked} \mapsto 1}^{\mathsf{r},\mathsf{t},(\mathtt{this},\mathsf{r})}_{\mathsf{I(R)}}$
        $* \ [\textsc{Lock}]^{\mathsf{r}}_{\pi} * [\tau_1]^{\mathsf{r}}_{\pi} * [\tau_2]^{\mathsf{r}}_{\pi}\}$
  x = CAS(this.locked, 0, 1);
    $\{(\mathsf{x} = \mathtt{this} * \boxed{\mathtt{this}.\mathsf{locked} \mapsto 1}^{\mathsf{r},\mathsf{t},(\mathtt{this},\mathsf{r})}_{\mathsf{I(R)}} * [\textsc{Lock}]^{\mathsf{r}}_{\pi} * [\tau_1]^{\mathsf{r}}_{\pi} * [\tau_2]^{\mathsf{r}}_{\pi} * [\textsc{Unlock}]^{\mathsf{r}}_{1} * \mathsf{R}) \ \vee$
        $(\mathsf{x} = \mathsf{null} * \boxed{\mathtt{this}.\mathsf{locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{\mathsf{r}}_{1}}^{\mathsf{r},\mathsf{t},(\mathtt{this},\mathsf{r})}_{\mathsf{I(R)}} * [\textsc{Lock}]^{\mathsf{r}}_{\pi} * [\tau_1]^{\mathsf{r}}_{\pi} * [\tau_2]^{\mathsf{r}}_{\pi})\}$
    $\{(\mathsf{x} = \mathtt{this} * \boxed{\mathtt{this}.\mathsf{locked} \mapsto 1}^{\mathsf{r},\mathsf{t},(\mathtt{this},\mathsf{r})}_{\mathsf{I(R)}} * [\textsc{Lock}]^{\mathsf{r}}_{\pi} * [\tau_1]^{\mathsf{r}}_{\pi} * [\tau_2]^{\mathsf{r}}_{\pi} * [\textsc{Unlock}]^{\mathsf{r}}_{1} * \mathsf{R}) \ \vee$
        $(\mathsf{x} = \mathsf{null} * \boxed{(\mathtt{this}.\mathsf{locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{\mathsf{r}}_{1}) \vee \mathtt{this}.\mathsf{locked} \mapsto 1}^{\mathsf{r},\mathsf{t},(\mathtt{this},\mathsf{r})}_{\mathsf{I(R)}}$
            $* \ [\textsc{Lock}]^{\mathsf{r}}_{\pi} * [\tau_1]^{\mathsf{r}}_{\pi} * [\tau_2]^{\mathsf{r}}_{\pi})\}$
  if(x == null) {
      $\{\boxed{(\mathtt{this}.\mathsf{locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{\mathsf{r}}_{1}) \vee \mathtt{this}.\mathsf{locked} \mapsto 1}^{\mathsf{r},\mathsf{t},(\mathtt{this},\mathsf{r})}_{\mathsf{I(R)}} * [\textsc{Lock}]^{\mathsf{r}}_{\pi} * [\tau_1]^{\mathsf{r}}_{\pi} * [\tau_2]^{\mathsf{r}}_{\pi}\}$
      $\{\mathsf{isLock}(\mathsf{t}, \mathsf{R}, \mathtt{this})\}$
    Acquire();
      $\{\mathsf{locked}(\mathsf{t}, \mathsf{R}, \mathtt{this}) * \mathsf{R}\}$
  } else {
      $\{\boxed{\mathtt{this}.\mathsf{locked} \mapsto 1}^{\mathsf{r},\mathsf{t},(\mathtt{this},\mathsf{r})}_{\mathsf{I(R)}} * [\textsc{Lock}]^{\mathsf{r}}_{\pi} * [\tau_1]^{\mathsf{r}}_{\pi} * [\tau_2]^{\mathsf{r}}_{\pi} * [\textsc{Unlock}]^{\mathsf{r}}_{1} * \mathsf{R}\}$
      $\{\mathsf{locked}(\mathsf{t}, \mathsf{R}, \mathtt{this}) * \mathsf{R}\}$
  }
    $\{\mathsf{locked}(\mathsf{t}, \mathsf{R}, \mathtt{this}) * \mathsf{R}\}$
}
```

The main step is the atomic compare-and-swap which accesses the locked field, which is owned by the shared lock region. Note that the immediate post-condition of the compare-and-swap is *not* stable. In particular, if the compare-and-swap fails, then in the instant the compare-and-swap fails, the lock is already locked. However, the owner of the lock is free to unlock the lock, hence it is not stable to assert the lock is locked. We thus immediately apply the rule of consequence, to weaken the post-condition to something that *is* stable.

The next step is thus to verify the atomic update of the shared region:

$$(\mathsf{x}).\ \{\boxed{(\mathtt{this}.\mathsf{locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{\mathsf{r}}_{1}) \vee \mathtt{this}.\mathsf{locked} \mapsto 1}^{\mathsf{r},\mathsf{t},(\mathtt{this},\mathsf{r})}_{\mathsf{I}(\mathsf{R})} * [\textsc{Lock}]^{\mathsf{r}}_{\pi} * [\tau_1]^{\mathsf{r}}_{\pi} * [\tau_2]^{\mathsf{r}}_{\pi}\}$$

$$\langle \mathsf{x} = \mathtt{CAS}(\mathtt{this}.\mathsf{locked}, 0, 1)\rangle$$

$$\left\{\begin{array}{l} (\mathsf{x} = \mathtt{this} * \boxed{\mathtt{this}.\mathsf{locked} \mapsto 1}^{\mathsf{r},\mathsf{t},(\mathtt{this},\mathsf{r})}_{\mathsf{I}(\mathsf{R})} * [\textsc{Lock}]^{\mathsf{r}}_{\pi} * [\tau_1]^{\mathsf{r}}_{\pi} * [\tau_2]^{\mathsf{r}}_{\pi} * [\textsc{Unlock}]^{\mathsf{r}}_{1} * \mathsf{R}) \vee \\ (\mathsf{x} = \mathsf{null} * \boxed{\mathtt{this}.\mathsf{locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{\mathsf{r}}_{1}}^{\mathsf{r},\mathsf{t},(\mathtt{this},\mathsf{r})}_{\mathsf{I}(\mathsf{R})} * [\textsc{Lock}]^{\mathsf{r}}_{\pi} * [\tau_1]^{\mathsf{r}}_{\pi} * [\tau_2]^{\mathsf{r}}_{\pi}) \end{array}\right\}$$

By rule OpenA, this reduces to proving that the compare-and-swap performs the state update on the shared region:

$$(\mathsf{x}).\ \{((\mathtt{this}.\mathsf{locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{\mathsf{r}}_{1}) \vee \mathtt{this}.\mathsf{locked} \mapsto 1) * [\textsc{Lock}]^{\mathsf{r}}_{\pi} * [\tau_1]^{\mathsf{r}}_{\pi} * [\tau_2]^{\mathsf{r}}_{\pi}\}$$

$$\langle \mathsf{x} = \mathtt{CAS}(\mathtt{this}.\mathsf{locked}, 0, 1)\rangle^{\mathsf{t}}$$

$$\left\{\begin{array}{l} (\mathsf{x} = \mathtt{this} * \mathtt{this}.\mathsf{locked} \mapsto 1 * [\textsc{Lock}]^{\mathsf{r}}_{\pi} * [\tau_1]^{\mathsf{r}}_{\pi} * [\tau_2]^{\mathsf{r}}_{\pi} * [\textsc{Unlock}]^{\mathsf{r}}_{1} * \mathsf{R}) \vee \\ (\mathsf{x} = \mathsf{null} * \mathtt{this}.\mathsf{locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{\mathsf{r}}_{1} * [\textsc{Lock}]^{\mathsf{r}}_{\pi} * [\tau_1]^{\mathsf{r}}_{\pi} * [\tau_2]^{\mathsf{r}}_{\pi}) \end{array}\right\}$$

and that this update is allowed:

$$\boxed{(\mathtt{this}.\mathsf{locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{\mathsf{r}}_{1}) \vee \mathtt{this}.\mathsf{locked} \mapsto 1}^{\mathsf{r},\mathsf{t},(\mathtt{this},\mathsf{r})}_{\mathsf{I}(\mathsf{R})} * [\textsc{Lock}]^{\mathsf{r}}_{\pi} * [\tau_1]^{\mathsf{r}}_{\pi} * [\tau_2]^{\mathsf{r}}_{\pi}$$

$$\rightsquigarrow^{\mathsf{r},\mathsf{t}}_{(\mathsf{x}).\langle \mathsf{x}=\mathtt{CAS}(\mathtt{this}.\mathsf{locked},0,1)\rangle}$$

$$(\mathsf{x} = \mathtt{this} * \boxed{\mathtt{this}.\mathsf{locked} \mapsto 1}^{\mathsf{r},\mathsf{t},(\mathtt{this},\mathsf{r})}_{\mathsf{I}(\mathsf{R})} * [\textsc{Lock}]^{\mathsf{r}}_{\pi} * [\tau_1]^{\mathsf{r}}_{\pi} * [\tau_2]^{\mathsf{r}}_{\pi} * [\textsc{Unlock}]^{\mathsf{r}}_{1} * \mathsf{R}) \vee$$

$$(\mathsf{x} = \mathsf{null} * \boxed{\mathtt{this}.\mathsf{locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{\mathsf{r}}_{1}}^{\mathsf{r},\mathsf{t},(\mathtt{this},\mathsf{r})}_{\mathsf{I}(\mathsf{R})} * [\textsc{Lock}]^{\mathsf{r}}_{\pi} * [\tau_1]^{\mathsf{r}}_{\pi} * [\tau_2]^{\mathsf{r}}_{\pi})$$

The first proof obligation follows by a fairly standard separation logic proof (using rule ACas). To prove that the update is allowed, it suffices (by the rule of consequence) to prove,

$$\boxed{(\mathtt{this}.\mathsf{locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{\mathsf{r}}_{1}) \vee \mathtt{this}.\mathsf{locked} \mapsto 1}^{\mathsf{r},\mathsf{t},(\mathtt{this},\mathsf{r})}_{\mathsf{I}(\mathsf{R})} * [\textsc{Lock}]^{\mathsf{r}}_{\pi} * [\tau_1]^{\mathsf{r}}_{\pi} * [\tau_2]^{\mathsf{r}}_{\pi}$$

$$\rightsquigarrow^{\mathsf{r},\mathsf{t}}_{(\mathsf{x}).\langle \mathsf{x}=\mathtt{CAS}(\mathtt{this}.\mathsf{locked},0,1)\rangle}$$

$$(\mathsf{x} = \mathtt{this} * \boxed{\mathtt{this}.\mathsf{locked} \mapsto 1}^{\mathsf{r},\mathsf{t},(\mathtt{this},\mathsf{r})}_{\mathsf{I}(\mathsf{R})}) \vee$$

$$(\mathsf{x} = \mathsf{null} * \boxed{\mathtt{this}.\mathsf{locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{\mathsf{r}}_{1}}^{\mathsf{r},\mathsf{t},(\mathtt{this},\mathsf{r})}_{\mathsf{I}(\mathsf{R})})$$

- From the update allowed rules, we can perform case analysis on each disjunction, obtaining the following four proof obligations:

$$\boxed{\mathtt{this}.\mathsf{locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{\mathsf{r}}_{1}}^{\mathsf{r},\mathsf{t},(\mathtt{this},\mathsf{r})}_{\mathsf{I}(\mathsf{R})} * [\textsc{Lock}]^{\mathsf{r}}_{\pi} * [\tau_1]^{\mathsf{r}}_{\pi} * [\tau_2]^{\mathsf{r}}_{\pi}$$

$$\rightsquigarrow^{\mathsf{r},\mathsf{t}}_{(\mathsf{x}).\langle \mathsf{x}=\mathtt{CAS}(\mathtt{this}.\mathsf{locked},0,1)\rangle} (\mathsf{x} = \mathtt{this} * \boxed{\mathtt{this}.\mathsf{locked} \mapsto 1}^{\mathsf{r},\mathsf{t},(\mathtt{this},\mathsf{r})}_{\mathsf{I}(\mathsf{R})})$$

$$\boxed{\mathtt{this}.\mathsf{locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{\mathsf{r}}_{1}}^{\mathsf{r},\mathsf{t},(\mathtt{this},\mathsf{r})}_{\mathsf{I}(\mathsf{R})} * [\textsc{Lock}]^{\mathsf{r}}_{\pi} * [\tau_1]^{\mathsf{r}}_{\pi} * [\tau_2]^{\mathsf{r}}_{\pi}$$

$$\rightsquigarrow^{\mathsf{r},\mathsf{t}}_{(\mathsf{x}).\langle \mathsf{x}=\mathtt{CAS}(\mathtt{this}.\mathsf{locked},0,1)\rangle} (\mathsf{x} = \mathsf{null} * \boxed{\mathtt{this}.\mathsf{locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{\mathsf{r}}_{1}}^{\mathsf{r},\mathsf{t},(\mathtt{this},\mathsf{r})}_{\mathsf{I}(\mathsf{R})})$$

$$\boxed{\mathtt{this}.\mathsf{locked} \mapsto 1}^{\mathsf{r},\mathsf{t},(\mathtt{this},\mathsf{r})}_{\mathsf{I}(\mathsf{R})} * [\textsc{Lock}]^{\mathsf{r}}_{\pi} * [\tau_1]^{\mathsf{r}}_{\pi} * [\tau_2]^{\mathsf{r}}_{\pi}$$

$$\rightsquigarrow^{\mathsf{r},\mathsf{t}}_{(\mathsf{x}).\langle \mathsf{x}=\mathtt{CAS}(\mathtt{this}.\mathsf{locked},0,1)\rangle} (\mathsf{x} = \mathtt{this} * \boxed{\mathtt{this}.\mathsf{locked} \mapsto 1}^{\mathsf{r},\mathsf{t},(\mathtt{this},\mathsf{r})}_{\mathsf{I}(\mathsf{R})})$$

$$\boxed{\mathtt{this}.\mathsf{locked} \mapsto 1}^{\mathsf{r},\mathsf{t},(\mathtt{this},\mathsf{r})}_{\mathsf{I}(\mathsf{R})} * [\textsc{Lock}]^{\mathsf{r}}_{\pi} * [\tau_1]^{\mathsf{r}}_{\pi} * [\tau_2]^{\mathsf{r}}_{\pi}$$

$$\rightsquigarrow^{\mathsf{r},\mathsf{t}}_{(\mathsf{x}).\langle \mathsf{x}=\mathtt{CAS}(\mathtt{this}.\mathsf{locked},0,1)\rangle} (\mathsf{x} = \mathsf{null} * \boxed{\mathtt{this}.\mathsf{locked} \mapsto 0 * \mathsf{R} * [\textsc{Unlock}]^{\mathsf{r}}_{1}}^{\mathsf{r},\mathsf{t},\mathtt{this}}_{\mathsf{I}(\mathsf{R})})$$

The first three proof obligations follow directly from rule AUAct, using the Lock, τ_1 and τ_2 action, respectively. The last obligation follows by rule AUFalse1, as a compare-and-swap from 0 to 1 cannot update locked from 1 to 0. In particular, by rule ACas it follows that

$$(\mathsf{x}).\{\mathtt{this}.\mathsf{locked} \mapsto 1\}\langle \mathsf{x} = \mathtt{CAS}(\mathtt{this}.\mathrm{locked}, 0, 1)\rangle\{\mathtt{this}.\mathsf{locked} \mapsto 1\}.$$

□

4 Model

In this section we define a model for our proof system and prove some meta-theoretic results about the model. The concrete interpretation of the logic is given in Section 5. The model and its meta-theory is defined using a standard classical meta logic.

The presentation of the model is strongly inspired by the Views framework [3] presentation. In the Views framework, the operational semantics of the underlying programming language operates on concrete machine states, while assertions are predicates on instrumented (abstract) machine states. The soundness of the logic ensures that any step at the concrete level has a corresponding step at the instrumented level. Our model can be seen as a concrete instance of the Views framework, instrumented with actions and protocols for modeling the CAP-part of the logic and a phantom heap for modeling phantom fields. In addition, everything is step-indexed to model the later modality, guarded recursion, and the embeddings between the logics. The Views framework features shared variable concurrency. Since mini C$^\sharp$ features fork-concurrency and threads with local stacks and a shared heap, we have generalized the Views framework with threads and thread-local state. To model guardedness, we model types as sets equipped with a step-indexed equivalence relation. Furthermore, to ensure modular reasoning about guardedness, we build non-expansiveness into the interpretation. The interpretation is thus given in the category of step-indexed equivalence relations and non-expansive functions, instead of the category of sets. We begin by defining the category of step-indexed equivalence relations and non-expansive functions.

Category of step-indexed equivalence relations $\boxed{\text{ASets} \in \text{Cats}}$

A step-indexed equivalence is a pair, $(X, (=_i)_{i\in\mathbb{N}})$, consisting of a set X and a set of equivalence relations, $=_i \subseteq X \times X$, indexed by natural numbers $i \in \mathbb{N}$. We require that the step-indexed equivalence relations become coarser at lower step-indicies:

$$\forall i \in \mathbb{N}.\ \forall x, y \in X.\ x =_{i+1} y \Rightarrow x =_i y$$

and that equality at every step-index corresponds to identity:

$$\forall x, y \in X.\ (\forall i \in \mathbb{N}.\ x =_i y) \Rightarrow x = y$$

Given two step-indexed equivalence relations

$$\mathcal{X} = (X, =_i^X) \in \text{ASets} \qquad\qquad \mathcal{Y} = (Y, =_i^Y) \in \text{ASets}$$

let $\mathcal{X} \to_{ne} \mathcal{Y}$ denote the set of non-expansive functions:

$$\mathcal{X} \to_{ne} \mathcal{Y} \stackrel{\text{def}}{=} \{f : X \to Y \mid \forall i \in \mathbb{N}.\ \forall x, y \in X.\ x =_i^X y \Rightarrow f(x) =_i^Y f(y)\}$$

Define ASets as the category of step-indexed equivalences and non-expansive functions. This category is equivalent to the category of bisected ultrametric spaces. Ultrametric spaces have previously been used to model guarded recursion [5, 2, 1]. We prefer this equivalent but more concrete presentation in terms of step-indexed equivalences.

Embedding $\boxed{\begin{array}{l}\Delta : \text{Sets} \to \text{ASets} \in \text{Cats} \\ U : \text{ASets} \to \text{Sets} \in \text{Cats}\end{array}}$

The category of sets embeds into ASets via Δ. Furthermore, Δ is a left-adjoint to the forgetful functor, U.

$$\Delta(X) \stackrel{\text{def}}{=} (X, (\{(x,x) \mid x \in X\})_{i\in\mathbb{N}}) \qquad U(X, (=_i)_{i\in\mathbb{N}}) \stackrel{\text{def}}{=} X$$
$$\Delta(f) \stackrel{\text{def}}{=} f \qquad U(f) \stackrel{\text{def}}{=} f$$

We will always write out Δ explicitly, when embedding sets into ASets; however, we will leave U implicit.

Cartesian closed structure $\boxed{1, \mathcal{X} \times \mathcal{Y}, \mathcal{X} \to \mathcal{Y} \in \text{ASets}}$

$$\begin{aligned} 1 &\stackrel{\text{def}}{=} (\{*\}, =_i^1) \\ \mathcal{X} \times \mathcal{Y} &\stackrel{\text{def}}{=} (X \times Y, =_i^{X\times Y}) \\ \mathcal{X} \to \mathcal{Y} &\stackrel{\text{def}}{=} (\mathcal{X} \to_{ne} \mathcal{Y}, =_i^{X\to Y}) \end{aligned}$$

where

$$\begin{array}{rcl} * =_i^1 * & \text{iff} & \top \\ (x_1, y_1) =_i^{X\times Y} (x_2, y_2) & \text{iff} & x_1 =_i^X x_2 \wedge y_1 =_i^Y y_2 \\ f =_i^{X\to Y} g & \text{iff} & \forall x \in X.\ f(x) =_i^Y g(x) \end{array}$$

for $\mathcal{X} = (X, =_i^X) \in$ ASets and $\mathcal{Y} = (Y, =_i^Y) \in$ ASets.

Semantic domains

The basic structure of the model is given by the following semantic domains. We assume countably infinite and disjoint sets of region identifiers (*RId*) action identifiers (*AId*) and region types (*RId*).

Instrumented states consist of three components: a local state, a shared state and an action model. The local state specifies the current local resources. Local resources consists of concrete fields, phantom fields and action permissions. The shared state is partitioned into regions. Each region consists of a local state, a region type and a protocol argument (of type *Val*). The local state component of a shared region specifies the local resources currently owned by that region. The action model associates parameterized protocols with region types. Parameterized protocols are represented as functions from action argument to actions. An action argument consists of a protocol argument, a region identifier and an action identifier. Lastly, actions are modeled as certain step-indexed relations on shared states. This avoids the circularity that would arise from defining actions as relations on shared states and action models, but it also means the model lacks support for general higher-order protocols. The model does however support state-independent protocols through the region type indirection. In particular, as shared states include region types and protocol arguments, actions *can* constrain the region types and protocol arguments of other regions.

$r \in RId$	region identifier
$a \in AId$	action identifier
$t \in RType$	region type
$PHeap \stackrel{\text{def}}{=} OId \times FName \stackrel{\text{fin}}{\rightharpoonup} Val$	phantom heap
$l \in LState \stackrel{\text{def}}{=} Heap \times PHeap \times Cap$	local state
$s \in SState \stackrel{\text{def}}{=} RId \stackrel{\text{fin}}{\rightharpoonup} (LState \times RType \times Val)$	shared state
$t \in Token \stackrel{\text{def}}{=} RId \times AId$	token
$c \in Cap \stackrel{\text{def}}{=} \{f \in Token \to [0,1] \mid dom_r(f) \text{ finite}\}$	capability map
$a \in Action \stackrel{\text{def}}{=} \{R \in \mathcal{P}(\mathbb{N} \times SState \times SState) \mid good(R)\}$	action
$AArg \stackrel{\text{def}}{=} Val \times RId \times AId$	action argument
$\varsigma \in AMod \stackrel{\text{def}}{=} RType \rightharpoonup (AArg \to Act)$	action model
$m \in \mathcal{M} \stackrel{\text{def}}{=} LState \times SState \times AMod$	instrumented states

We require that actions are good, meaning downwards-closed in the step-index, closed under allocation of new regions and closed under arbitrary changes to additional regions.

$$\begin{aligned}
good(R) \stackrel{\text{def}}{=}\ & \forall (i, s_1, s_2) \in R.\ \forall j \leq i.\ \forall r \in RId \setminus dom(s_2).\\
&\quad \forall t \in RType.\ \forall a \in Val.\ \forall l, l' \in LState.\\
&\qquad s_1 \leq s_2 \wedge (j, s_1, s_2) \in R \wedge (j, s_1, s_2[r \mapsto (l', t, a)]) \in R \wedge\\
&\qquad (j, s_1[r \mapsto (l, t, a)], s_2[r \mapsto (l', t, a)]) \in R\\
s_1 \leq s_2 \stackrel{\text{def}}{=}\ & dom(s_1) \subseteq dom(s_2) \wedge (\forall r \in dom(s_1).\ s_1(r).t = s_2(r).t \wedge s_1(r).a = s_2(r).a)\\
dom_r(f) \stackrel{\text{def}}{=}\ & \{r \in RId \mid \exists \alpha \in AId.\ f(r, \alpha) > 0\}
\end{aligned}$$

We use $m.l$, $m.s$ and $m.a$ as shorthand to refer to the local state, shared state and action model component of an instrumented state m, respectively. We use $s(r).l$, $s(r).t$ and $s(r).a$ to refer to the local state, region type and protocol parameter component of a shared region r from the shared state s. We use $l.h$, $l.p$, and $l.c$ to refer to the heap, the phantom heap and the capability map of a local state $l \in LState$.

Values $\boxed{Val \in \text{Sets}}$

Let Val denote the least set such that

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

Region types $\boxed{\mathit{RType} \in \text{Sets}}$

Region types are simply modeled as strings, with the prefix-ordering. Values thus include region types.

$$\mathit{RType} = \mathit{Strings}$$

We use t^* as notation for $\{s \in \mathit{RType} \mid t \text{ is a prefix of } s\}$ when $t \in \mathit{RType}$.

Composition operators $\boxed{\begin{array}{r} \bullet_{\mathit{LState}} : \mathit{LState} \times \mathit{LState} \rightharpoonup \mathit{LState} \\ \bullet_{\mathcal{M}} : \mathcal{M} \times \mathcal{M} \rightharpoonup \mathcal{M} \end{array}}$

Instrumented and local states compose using the partial $\bullet_{\mathit{LState}}$ and $\bullet_{\mathcal{M}}$ operators. Undefinedness indicates the two states are incompatible and do not compose. For instance, two local states that assert ownership of the same field are incompatible and do not compose.

$$X \bullet_{\uplus} Y \overset{\text{def}}{=} \begin{cases} X \cup Y & \text{if } X \cap Y = \emptyset \\ \text{undef} & \text{otherwise} \end{cases}$$

$$x \bullet_{=} y \overset{\text{def}}{=} \begin{cases} x & \text{if } x = y \\ \text{undef} & \text{otherwise} \end{cases}$$

$$f \bullet_{+} g \overset{\text{def}}{=} \begin{cases} \lambda x.\ f(x) + g(x) & \text{if } \forall x.\ f(x) + g(x) \leq 1 \\ \text{undef} & \text{otherwise} \end{cases}$$

$$(oh_1, th_1, ch_1) \bullet_{\mathit{Heap}} (oh_2, th_2, ch_2) \overset{\text{def}}{=} (oh_1 \bullet_{\uplus} oh_2, th_1 \bullet_{=} th_2, ch_1 \bullet_{=} ch_2)$$

$$(h_1, ph_1, c_1) \bullet_{\mathit{LState}} (h_2, ph_2, c_2) \overset{\text{def}}{=} (h_1 \bullet_{\mathit{Heap}} h_2, ph_1 \bullet_{\uplus} ph_2, c_1 \bullet_{+} c_2)$$

$$(l_1, s_1, a_1) \bullet_{\mathcal{M}} (l_2, s_2, a_2) \overset{\text{def}}{=} (l_1 \bullet_{\mathit{LState}} l_2, s_1 \bullet_{=} s_2, a_1 \bullet_{=} a_2)$$

Extension ordering $\boxed{\leq_{\mathcal{M}} : \mathcal{M} \times \mathcal{M} \to 2}$

$$m_1 \leq_{\mathcal{M}} m_2 \quad \text{iff} \quad \exists m \in \mathcal{M}.\ m_2 = m_1 \bullet m$$

The extension ordering induced by $\bullet$ defines a partial order.

View $\boxed{\mathcal{V} \in \text{Sets}}$

We interpret propositions in the assertion logic as certain step-indexed predicates on instrumented states, called Views. In particular, we require that views are downwards-closed in the step-index, closed under allocation of new regions and protocols, and upwards-closed in the local state and the local states of shared regions.

$$
\begin{aligned}
p, q \in \mathcal{V} \overset{\text{def}}{=} \{p \in \mathcal{P}(\mathbb{N} \times \mathcal{M}) \mid \\
&(\forall (i, m) \in p.\ \forall j \leq i.\ \forall n \geq_{\mathcal{M}} m.\ (j, n) \in p) \land \\
&(\forall (i, m_1) \in p.\ \forall m_2 \in \mathcal{M}.\ m_1 =_i^{\mathcal{M}} m_2 \Rightarrow (i, m_2) \in p) \land \\
&(\forall (i, (l, s, \varsigma)) \in p.\ \forall r \in (\mathit{RId} \setminus \mathit{dom}(s)).\ \forall t \in \mathit{RType}.\ \forall l_r \in \mathit{LState}.\ \forall a \in \mathit{Val}. \\
&\quad (i, (l, s[r \mapsto (l_r, t, a)], \varsigma)) \in p) \land \\
&(\forall (i, (l, s, \varsigma)) \in p.\ \forall r \in (\mathit{RType} \setminus \mathit{dom}(\varsigma)).\ \forall I \in \mathit{AArg} \to \mathit{Act}. \\
&\quad (i, (l, s, \varsigma[r \mapsto I])) \in p)\}
\end{aligned}
$$

where $=_i^{\mathcal{M}} \subseteq \mathcal{M} \times \mathcal{M}$ is given by

$$
\begin{aligned}
(l_1, s_1, \varsigma_1) =_i^{\mathcal{M}} (l_2, s_2, \varsigma_2) \quad \text{iff} \quad & l_1 = l_2 \land s_1 = s_2 \land \mathit{dom}(\varsigma_1) = \mathit{dom}(\varsigma_2) \land \\
& (\forall r \in \mathit{dom}(\varsigma_1).\ \forall \alpha \in \mathit{AArg}.\ \varsigma_1(r)(\alpha)|_i = \varsigma_2(r)(\alpha)|_i)
\end{aligned}
$$

and $R|_i$ is given by

$$R|_i = \{(j, s_1, s_2) \in \mathbb{N} \times \mathit{SState} \times \mathit{SState} \mid j \leq i \land (j, s_1, s_2) \in R\}$$

Assertions $\boxed{\mathit{Prop} \in \text{ASets}}$

We consider two semantic assertion propositions i-equivalent if they agree on all instrumented states for step-indexes strictly smaller than i.

$$\mathit{Prop} \overset{\text{def}}{=} (\mathcal{V}, (=_i^{\mathcal{V}})_{i \in \mathbb{N}})$$

where $=_i^{\mathcal{V}} \subseteq \mathcal{V} \times \mathcal{V}$ is given by

$$p =_i^{\mathcal{V}} q \quad \text{iff} \quad \forall j < i.\ \forall m \in \mathcal{M}.\ (j, m) \in p \Leftrightarrow (j, m) \in q$$

Specification $\boxed{\mathit{Spec} \in \text{ASets}}$

Likewise, we consider two semantic specification propositions i-equivalent if they agree at step-indexes strictly smaller than i.

$$\mathit{Spec} \overset{\text{def}}{=} (\mathsf{P}^{\downarrow}(\mathbb{N}), (=_i^{\mathsf{P}^{\downarrow}(\mathbb{N})})_{i \in \mathbb{N}})$$

where $=_i^{\mathcal{P}^{\downarrow}(\mathbb{N})} \subseteq \mathcal{P}^{\downarrow}(\mathbb{N}) \times \mathcal{P}^{\downarrow}(\mathbb{N})$ is given by

$$p =_i^{\mathcal{P}^{\downarrow}(\mathbb{N})} q \quad \text{iff} \quad \forall j < i.\ j \in p \Leftrightarrow j \in q$$

Region collapse $\boxed{\ulcorner - \urcorner : \mathit{LState} \times \mathit{SState} \rightharpoonup \mathit{LState}}$

$$\ulcorner (l, s) \urcorner \overset{\text{def}}{=} l \bullet_{\mathit{LState}} \Pi_{r \in \mathit{dom}(s)} \pi_1(s(r))$$

Action model extension $\boxed{(-) \leq (=) : \mathcal{P}(\mathit{AMod} \times \mathit{AMod})}$

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

Shared state restriction $\boxed{(-)|_{(=)} : \mathit{SState} \times \mathcal{P}(\mathit{RType}) \to \mathit{SState}}$

$$s|_A(r) \stackrel{\text{def}}{=} \begin{cases} s(r) & \text{if } r \in \mathit{dom}(s) \text{ and } s(r).t \in A \\ \text{undef} & \text{otherwise} \end{cases}$$

Interference relation $\boxed{\hat{R}^{(-)}_{(=)}, R^{(-)}_{(=)} : \mathcal{P}(\mathit{RType}) \times \mathbb{N} \to \mathcal{P}(\mathcal{M} \times \mathcal{M})}$

The interference relation R^A_i describes possible interference from the environment. It is defined as the reflexive, transitive closure of the single-action interference relation, $\hat{R}^A_i$, that describes possible environment interference using at most one action on each region. This thus forces a common granularity on synchronized actions. In addition to the step-index i, the single-action interference relation is also indexed by a set of region types A. This is the types of regions that are allowed to change, and of regions that actions justifying those changes are allowed to depend upon.

$$R^A_i \stackrel{\text{def}}{=} (\hat{R}^A_i)^*$$

where $(-)^*$ denotes the reflexive, transitive closure operator on binary relations, and

$$\begin{aligned}
&(l_1, s_1, \varsigma_1)\ \hat{R}^A_i\ (l_2, s_2, \varsigma_2) \quad \text{iff} \\
&\quad l_1 = l_2 \wedge s_1 \leq s_2 \wedge \varsigma_1 \leq \varsigma_2 \wedge \\
&\qquad \exists c \in \mathit{Cap}.\ c = \pi_c(\ulcorner (l_1, s_1) \urcorner). \\
&\qquad\quad (\forall r \in \mathit{dom}(s_1).\ s_1(r) = s_2(r) \vee \\
&\qquad\qquad (\exists \alpha \in \mathit{AId}.\ s_1(r).t \in A \wedge c(r, \alpha) < 1 \\
&\qquad\qquad\quad \wedge\ (i, s_1|_A, s_2|_A) \in \varsigma_1(s_1(r).t)(s_1(r).a, r, \alpha)))
\end{aligned}$$

Stability $\boxed{\mathit{stable} : \mathit{Prop} \to \mathit{Spec} \in \text{ASets}}$

An assertion is stable at step-index i if it is closed under R^{RType}_i.

$$\mathit{stable}(p) \stackrel{\text{def}}{=} \{i \in \mathbb{N} \mid \forall j \leq i.\ \forall (m_1, m_2) \in R^{\mathit{RType}}_j.\ (j, m_1) \in p \Rightarrow (j, m_2) \in p\}$$

We also use *stable* as notation for the point-wise lifting of *stable* to predicates. Thus, $\mathit{stable}(p)$ is notation for $\{i \in \mathbb{N} \mid \forall x \in X.\ i \in \mathit{stable}(p(x))\}$ when $p \in X \to \mathcal{V}$.

For *stable* to be a well-defined morphism in ASets, we have to prove that *stable* is non-expansive. To illustrate, we write out the proof of non-expansiveness of *stable*; however, in general we will omit trivial non-expansiveness proofs.

Lemma 4. *stable is non-expansive:*

$$\forall p, q \in \mathcal{V}.\ \forall i \in \mathbb{N}.\ p =_i^{\mathcal{V}} q \Rightarrow \mathit{stable}(p) =_i^{\mathcal{P}^{\downarrow}(\mathbb{N})} \mathit{stable}(q)$$

Proof.

- let $j < i$ and assume $j \in \mathit{stable}(p)$
- let $k \leq j$, $(m_1, m_2) \in R_k^{\mathit{RType}}$, and $(k, m_1) \in q$
- then $(k, m_1) \in p$ and thus, as $j \in \mathit{stable}(p)$, $(k, m_2) \in p$
- thus, $(k, m_2) \in q$

□

Erasure $\boxed{\lfloor - \rfloor : \mathcal{M} \rightharpoonup \mathit{Heap}}$

The erasure function, $\lfloor - \rfloor$, erases the instrumentation from instrumented states, yielding a concrete state. Since CAP partitions the state into local and shared states, the erasure first collapses the local and shared states. The erasure is then the heap component of the collapsed local states. The erasure function further requires that the phantom heap does not contain any phantom fields of objects that have not been allocated yet. This allows us to allocate new phantom fields in constructors.

$$\lfloor (l, s, \varsigma) \rfloor \stackrel{\text{def}}{=} \begin{cases} h & \text{if } (h, ph, c) = \ulcorner (l, s) \urcorner \text{ and } \pi_1(\mathit{dom}(ph)) \subseteq \mathit{objs}(h) \\ \text{undef} & \text{otherwise} \end{cases}$$

where $\mathit{objs}(oh, th, ch) = \mathit{dom}(th)$.

Step-indexed erasure $\boxed{\lfloor - \rfloor_{(=)} : \mathcal{V} \times \mathbb{N} \to \mathcal{P}(\mathit{Heap})}$

$$\lfloor p \rfloor_i \stackrel{\text{def}}{=} \lfloor \{ m \in \mathcal{M} \mid (i, m) \in p \} \rfloor$$

View-shift $\boxed{\sqsubseteq : \mathit{Prop} \times \mathit{Prop} \to \mathit{Spec} \in \text{ASets}}$

Intuitively, a view-shift from p to q expresses that there exists a step at the instrumented level from p to q, corresponding to a no-op at the concrete level.

$$p \sqsubseteq q \stackrel{\text{def}}{=} \{ i \in \mathbb{N} \mid \forall r \in \mathcal{V}.\ \forall j \in \mathbb{N}.\ 0 \leq j \leq i \wedge j \in \mathit{stable}(r) \Rightarrow \lfloor p * r \rfloor_j \subseteq \lfloor q * r \rfloor_j \}$$

We also use $\sqsubseteq$ as notation for the point-wise lifting of $\sqsubseteq$ to predicates. Thus, $p \sqsubseteq q$ is notation for $\{ i \in \mathbb{N} \mid \forall x \in X.\ i \in (p(x) \sqsubseteq q(x)) \}$ when $p, q \in X \to \mathcal{V}$.

Atomic action $\boxed{\mathit{sat} : \Delta(\mathit{Act}) \times \mathit{Prop} \times \mathit{Prop} \to \mathit{Spec} \in \text{ASets}}$

Likewise, an atomic action a from p to q expresses that there exists a step at the instrumented level from p to q, corresponding to the atomic action a at the concrete level.

$$\begin{aligned} a\ \mathit{sat}\ \{p\}\{q\} \overset{\text{def}}{=} \{i \in \mathbb{N} \mid \forall r \in \mathcal{V}.\ \forall j \in \mathbb{N}.& \\ 1 \le j \le i \wedge j \in \mathit{stable}(r) \Rightarrow [\![a]\!](\lfloor p * r \rfloor_j) \subseteq \lfloor q * r \rfloor_{j-1}\}& \end{aligned}$$

Thread safety $\boxed{\mathit{safe}_i : \mathit{Thread} \times \mathcal{V} \times (\mathit{Stack} \to \mathcal{V}) \to 2}$

Informally, safety is supposed to establish a simulation between the concrete level and the instrumented level, such that every step at the concrete level has a corresponding step at the instrumented level.

Safety is expressed in terms of a single thread, as possible interference from the environment is implicitly given by the pre-condition and post-condition through the interference relation. Formally, this is captured by Theorem 1, which relates the execution of an entire thread pool, to the safety of the individual threads.

$$\begin{aligned} \mathit{safe}_0(x,p,q) &\overset{\text{def}}{=} \top \\ \mathit{safe}_{i+1}(x,p,q) &\overset{\text{def}}{=} (\mathit{irr}(x) \wedge i+1 \in (p \sqsubseteq q(x_l))) \vee \\ &\quad (\forall T \in \mathit{TPool}.\ \forall a \in \mathit{Act}.\ \forall y \in \mathit{Thread}.\ x \xrightarrow{a} \{y\} \uplus T \wedge x.t = y.t \Rightarrow \\ &\qquad \exists p' \in \{y\} \uplus T \to \mathcal{V}. \\ &\qquad\quad (\forall z \in \{y\} \uplus T.\ i+1 \in \mathit{stable}(p'(z))) \wedge \\ &\qquad\quad i+1 \in (a\ \mathit{sat}\ \{p\}\{p'(y) * \circledast_{z \in T} p'(z)\}) \wedge \\ &\qquad\quad \mathit{safe}_i(y, p'(y), q) \wedge \\ &\qquad\quad \forall z \in T.\ \mathit{safe}_i(z, p'(z), \lambda l'.\ \top)) \end{aligned}$$

If a thread x is irreducible (i.e., x has terminate), then x is safe relative to p and q if there exists a step at the instrumented level from p to $q(x.l)$, corresponding to a no-op at the concrete level.

If a thread x can take a single step to y ($x.t = y.t$ ensures x and y have the same thread id) with action a by spawning threads T, then x is safe relative to p and q if:

- there exists stable predicates $p'(z)$ describing the intermediate state (at the instrumented level) of each thread $z \in \{y\} \uplus T$
- such that it is possible to take a step at the instrumented level from p to $\circledast_{z \in \{y\} \uplus T} p'(z)$ (thus splitting the combined intermediate resources between each thread), corresponding to the atomic action a at the concrete level
- and y is safe relative to $p'(y)$ and q
- and all spawned threads $z \in T$ are safe relative to $p'(z)$ and any post-condition

Since the post-condition q only describes the the terminal state of the initial thread x, only y (the thread x after one step of execution) is required to be safe relative to q.

Thread safety $\boxed{\mathit{safe} : \Delta(\mathit{Thread}) \times \mathit{Prop} \times (\Delta(\mathit{Stack}) \to \mathit{Prop}) \to \mathit{Spec} \in \text{ASets}}$

$$\mathit{safe}(x, p, q) \stackrel{\text{def}}{=} \{i \in \mathbb{N} \mid \mathit{safe}_i(x, p, q)\}$$

Statement safety $\boxed{\mathit{safe} : \Delta(\mathit{seq}\ \mathit{Stm}) \times (\Delta(\mathit{Stack}) \to \mathit{Prop})^2 \to \mathit{Spec} \in \text{ASets}}$

$$\mathit{safe}(s, p, q) \stackrel{\text{def}}{=} \{i \in \mathbb{N} \mid \forall t \in \mathit{TId}.\ \forall l \in \mathit{Stack}.\ \mathit{safe}_i((t, l, \mathit{stm}(s)), p(l), q)\}$$

4.1 Views meta-theory

In this section we develop some of the standard Views meta-theory for our step-indexed multithreaded safety predicate, including the rule of consequence (Lemma 13), the frame rule (Lemma 14), and sequential composition (Lemma 16). The main result is Theorem 1, which relates the execution of a thread pool with the safety of the individual threads.

Separation logic connectives

$$\boxed{\begin{aligned} emp &: Prop \\ * &: Prop \times Prop \to Prop \in \text{ASets}\end{aligned}}$$

$$\begin{aligned} emp &\stackrel{\text{def}}{=} \mathbb{N} \times \mathcal{M} \\ p * q &\stackrel{\text{def}}{=} \{(i,m) \in \mathbb{N} \times \mathcal{M} \mid \exists m_1, m_2 \in \mathcal{M}.\ m = m_1 \bullet_{\mathcal{M}} m_2 \wedge (i,m_1) \in p \wedge (i,m_2) \in q\}\end{aligned}$$

Image of an interference relation

$$\boxed{(-)(=) : \mathcal{P}(\mathcal{V} \times \mathcal{V}) \times \mathcal{V} \to \mathcal{V}}$$

$$R(p) \stackrel{\text{def}}{=} \{(i,m') \in \mathbb{N} \times \mathcal{M} \mid (i,m) \in p \wedge m\ R\ m'\}$$

Lemma 5.

$$\begin{aligned} &\forall i \in \mathbb{N}.\ \forall A \in \mathcal{P}(\mathit{RType}). \\ &\qquad (\forall p, q \in \mathcal{V}.\ R_i^A(p * q) \subseteq R_i^A(p) * R_i^A(q)) \wedge (R_i^A(emp) \subseteq emp)\end{aligned}$$

Lemma 6.

$$\forall i \in \mathbb{N}.\ \forall j \leq i.\ \forall A \in \mathcal{P}(\mathit{RType}).\ R_i^A \subseteq R_j^A$$

Lemma 7.

$$\forall i \in \mathbb{N}.\ \forall A_1, A_2 \in \mathcal{P}(\mathit{RType}).\ A_1 \subseteq A_2 \Rightarrow \hat{R}_i^{A_1} \subseteq \hat{R}_i^{A_2}$$

Lemma 8.

$$\forall i \in \mathbb{N}.\ \forall A \in \mathcal{P}(\mathit{RType}).\ \forall p, q \in \mathcal{V}.\ p \subseteq q \Rightarrow i \in (p \sqsubseteq^A q)$$

Lemma 9.

$$\forall p, q, r \in \mathcal{V}.\ (p \sqsubseteq q) \cap \mathit{stable}(r) \subseteq (p * r \sqsubseteq q * r)$$

Lemma 10.

$$\forall a \in \mathit{Act}.\ \forall p, q, r \in \mathcal{V}.\ (a\ \mathit{sat}\ \{p\}\{q\}) \cap \mathit{stable}(r) \subseteq (a\ \mathit{sat}\ \{p * r\}\{q * r\})$$

Lemma 11.

$$\begin{aligned} &\forall A \in \mathcal{P}(\mathit{RType}).\ \forall a \in \mathit{Act}.\ \forall p, q, r \in \mathcal{V}. \\ &\qquad (a\ \mathit{sat}^A\ \{p\}\{q\}) \cap \mathit{stable}(r) \subseteq (a\ \mathit{sat}^A\ \{p * r\}\{q * r\})\end{aligned}$$

Lemma 12 (Basic $\sqsubseteq$-closure)**.**

$$\forall i \in \mathbb{N}.\ \forall a \in \mathit{Act}.\ \forall p, p', q', q \in \mathcal{V}.$$
$$i+1 \in (p \sqsubseteq p') \wedge i+1 \in (a\ \mathit{sat}\ \{p'\}\{q'\}) \wedge i \in (q' \sqsubseteq q) \Rightarrow i+1 \in (a\ \mathit{sat}\ \{p\}\{q\})$$

Proof.

- let $r \in \mathcal{V}$ and $1 \leq j \leq i+1$
- then
$$\lfloor p * r \rfloor_j \subseteq \lfloor p' * r \rfloor_j \qquad [\![a]\!](\lfloor p' * r \rfloor_j) \subseteq \lfloor q' * r \rfloor_{j-1} \qquad \lfloor q' * r \rfloor_{j-1} \subseteq \lfloor q * r \rfloor_{j-1}$$
- and thus
$$[\![a]\!](\lfloor p * r \rfloor_j) \subseteq \lfloor q * r \rfloor_{j-1}$$

□

Lemma 13 (Consequence)**.**

$$\forall i \in \mathbb{N}.\ \forall x \in \mathit{Thread}.\ \forall p, p' \in \mathcal{V}.\ \forall q', q \in \mathit{Stack} \to \mathcal{V}.$$
$$i \in (p \sqsubseteq p') \wedge i \in \mathit{safe}(x, p', q') \wedge i \in (q' \sqsubseteq q) \Rightarrow i \in \mathit{safe}(x, p, q)$$

Proof. By induction on i.

- Case $i = 0$: trivial.
- Case $i = j + 1$:
  - if $irr(x)$ then $j+1 \in (p \sqsubseteq p') \cap (p' \sqsubseteq q'(x_l)) \cap (q'(x_l) \sqsubseteq q(x_l))$
  - otherwise, suppose $x \xrightarrow{a} \{y\} \uplus T$
  - by safety there thus exists
  $$p'' : \{y\} \uplus T \to \mathcal{V}$$
  such that
  $$\forall z \in \{y\} \uplus T.\ j+1 \in \mathit{stable}(p''(z))$$
  $$j+1 \in (a\ \mathit{sat}\ \{p'\}\{p''(y) * (\circledast_{z \in T} p''(z))\})$$
  $$\mathit{safe}_j(y, p''(y), q')$$
  $$\forall z \in T.\ \mathit{safe}_j(z, p''(z), \lambda\_.\ \top)$$
  - hence, by Lemma 12,
  $$j+1 \in (a\ \mathit{sat}\ \{p\}\{p''(y) * (\circledast_{x \in T} p''(x))\})$$

– and by the induction hypothesis,

$$\mathit{safe}_j(y, p''(y), q)$$

□

Lemma 14 (Frame)**.**

$$\forall x \in \mathit{Thread}.\ \forall p, r \in \mathcal{V}.\ \forall q \in \mathit{Stack} \to \mathcal{V}.$$
$$\mathit{safe}(x, p, q) \cap \mathit{stable}(r) \subseteq \mathit{safe}(x, p * r, \lambda l'.\ q(l') * r)$$

Proof. By induction on the step-index.

- Case $i = 0$: trivial.
- Case $i = j + 1$:
  - if $\mathit{irr}(x)$ then $i \in (p \sqsubseteq q(x_l))$ and thus $i \in (p * r \sqsubseteq q(x_l) * r)$
  - otherwise, suppose $x \xrightarrow{a} \{y\} \uplus T$
  - by safety there thus exists

    $$p' : \{y\} \uplus T \to \mathcal{V}$$

    such that

    $$\begin{array}{l} \forall z \in \{y\} \uplus T.\ i \in \mathit{stable}(p''(z)) \\ i \in (a\ \mathit{sat}\ \{p\}\{p'(y) * (\circledast_{z \in T} p'(z))\}) \\ \mathit{safe}_j(y, p'(y), q) \\ \forall z \in T.\ \mathit{safe}_j(z, p'(z), \lambda\_.\ \top) \end{array}$$

  - hence, by Lemma 10,

    $$i \in (a\ \mathit{sat}\ \{p * r\}\{p'(y) * r * (\circledast_{z \in T} p'(z))\})$$

  - and by the induction hypothesis,

    $$\mathit{safe}_j(y, p'(y) * r, \lambda l'.\ q(l') * r)$$

□

Lemma 15.

$$\forall t \in \mathit{TId}.\ \forall l, l' \in \mathit{LState}.\ \forall T \in \mathit{TPool}.\ \forall s_1, s_2, s_3 \in \mathit{TCStack}.\ \forall a \in \mathit{Act}.$$
$$s_1 \neq \varepsilon \wedge (t, l, s_1; s_2) \xrightarrow{a} \{(t, l', s_3)\} \uplus T \ \Rightarrow$$
$$(\exists s_4 \in \mathit{TCStack}.\ (t, l, s_1) \xrightarrow{a} \{(t, l', s_4)\} \uplus T \wedge s_3 = s_4; s_2)$$

Proof.

- Case SEQ:
  - by definition,

$$(l, hd(s_1)) \xrightarrow{a} (l', s'_1), \qquad s_3 = s'_1; tail(s_1; s_2), \qquad T = \emptyset$$

  - hence, by SEQ,

$$(t, l, s_1) \xrightarrow{a} \{(t, l', s'_1; tail(s_1))\}$$

  - take $s_4 = s_1; tail(s_1)$
- Case FORK:
  - by definition there exists an l_d and s_d such that

$$l' = l, \qquad s_3 = tail(s_1; s_2), \qquad T = \{(t', l_d, s_d)\}$$

  - hence, by FORK,

$$(t, l, s_1) \xrightarrow{a} \{(t, l', tail(s_1))\} \uplus T$$

  - take $s_4 = tail(s_1)$

□

Lemma 16 (Sequential composition)**.**

$$\forall i \in \mathbb{N}.\ \forall t \in \mathit{TId}.\ \forall l \in \mathit{Stack}.\ \forall s_1, s_2 \in \mathit{Stm}.\ \forall p \in \mathcal{V}.\ \forall q, r \in \mathit{Stack} \to \mathcal{V}.$$
$$i \in \mathit{safe}((t, l, s_1), p, q) \wedge i \in \mathit{safe}(s_2, q, r) \Rightarrow i \in \mathit{safe}((t, l, s_1; s_2), p, r)$$

Proof. By induction on i.

- Case $i = 0$: trivial.
- Case $i = j + 1$:
  - if $irr(s_1; s_2)$ then $irr(s_1)$ and $irr(s_2)$ and thus

$$i \in (p \sqsubseteq q(l)) \cap (q(l) \sqsubseteq r(l))$$

  - otherwise, $(t, l, s_1; s_2) \xrightarrow{a} \{(t, l', s_3)\} \uplus T$
  - Case $s_1 = \varepsilon$:
    * from $s_1 = \varepsilon$ it follows that $irr(s_1)$ and thus $i \in (p \sqsubseteq q(l))$
    * hence, by Lemma 13,

$$\mathit{safe}_i((t, l, s_2), p, r)$$

- Case $s_1 \neq \varepsilon$:
  - by Lemma 15 there exists an s_4 such that,
$$(t, l, s_1) \xrightarrow{a} \{(t, l', s_4)\} \uplus T, \qquad s_3 = s_4; s_2$$
  - let y denote (t, l', s_4)
  - by safety of s_1 there thus exists
$$p' \in \{y\} \uplus T \to \mathcal{V}$$
    such that
$$\begin{array}{l} \forall z \in \{y\} \uplus T.\ i \in \mathit{stable}(p'(z)) \\ a\ \mathit{sat}_i\ \{p\}\ \{p'(y) * \circledast_{z \in T} p'(z)\} \\ \mathit{safe}_j((t, l', s_4), p'(y), q) \\ \forall z \in T.\ \mathit{safe}_j(z, p'(z), \lambda\_.\ \top) \end{array}$$
  - by downwards-closure it follows that
$$\mathit{safe}_j(s_2, q, r)$$
  - and thus, by the induction hypothesis,
$$\mathit{safe}_j((t, l', s_4; s_2), p'(y), r)$$

□

Theorem 1 (Evaluation safety). *For any $i, j \in \mathbb{N}$, $T \in \mathit{TPool}$,*
$$p : T \to \mathcal{V}, \qquad q : T \to \mathit{Stack} \to \mathcal{V}, \qquad h \in \lfloor \circledast_{x \in T} p(x) \rfloor_j$$
if
$$i < j, \qquad \forall x \in T.\ \mathit{safe}_j(x, p(x), q(x)), \qquad (T, h) \to_i (T', h'), \qquad \mathit{irr}(T')$$
$$j \in \mathit{stable}(p) \qquad j \in \mathit{stable}(q)$$
then
$$h' \in \lfloor \circledast_{x \in T} q(x)(l_x) \rfloor_{j-i}$$
where
$$l_x = y.l, \qquad \text{if } y \in T' \text{ and } x.t = y.t$$

Proof. By induction on i.

- Case $i = 0$:

- since $i = 0$ it follows that $T' = T$ and $h' = h$
- hence $irr(x)$ for every $x \in T = T'$
- from safety it thus follows that $j \in (p(x) \sqsubseteq q(x)(x.l))$ for every $x \in T$
- hence, by stability of p and q,

$$j \in (\circledast_{x\in T} p(x) \sqsubseteq \circledast_{x\in T} q(x)(x.l))$$

  and thus

$$h' = h \in \lfloor \circledast_{x\in T} p(x) \rfloor_j \subseteq \lfloor \circledast_{x\in T} q(x)(x.l) \rfloor_j = \lfloor \circledast_{x\in T} q(x)(l_x) \rfloor_j$$

- Case $i = k + 1$:
  - suppose

$$(T, h) \to (T'', h'') \to_k (T', h')$$

  - there thus exists an $x \in T$ and $y \in T''$ such that $x.t = y.t$,

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

  - from the safety of x there thus exists

$$p' \in \{y\} \uplus T''' \to \mathcal{V}$$

  such that

$$\begin{array}{l} \forall z \in \{y\} \uplus T'''.\ j \in \mathit{stable}(p'(z)) \\ a\ \mathit{sat}_j\ \big\{p(x)\big\}\ \big\{p'(y) * (\circledast_{z\in T'''} p'(z))\big\} \\ \mathit{safe}_{j-1}(y, p'(y), q(x)) \\ \forall z \in T'''.\ \mathit{safe}_{j-1}(z, p'(z), \lambda l'.\ \top) \end{array}$$

  - let

$$\begin{array}{ll} p'' = [z \in (T \setminus \{x\}) \mapsto p(z), z \in \{y\} \uplus T''' \mapsto p'(z)] & : T'' \to \mathcal{V} \\ q'' = [y \mapsto q(x), z \in (T \setminus \{x\}) \mapsto q(z), z \in T''' \mapsto \lambda\_.\ \top] & : T'' \to \mathit{Stack} \to \mathcal{V} \end{array}$$

  - then $\forall z \in T''.\ \mathit{safe}_{j-1}(z, p''(z), q''(z))$
  - from Lemma 10 it follows that

$$a\ \mathit{sat}_j\ \big\{\circledast_{x\in T}\, p(x)\big\}\ \big\{p'(y) * (\circledast_{z\in T'''} p'(z)) * (\circledast_{z\in(T\setminus\{x\})} p(z))\big\}$$

  and thus

$$h'' \in \lfloor p'(y) * (\circledast_{z\in T'''} p'(z)) * (\circledast_{z\in(T\setminus\{x\})} p(z)) \rfloor_{j-1} = \lfloor \circledast_{x\in T''} p''(x) \rfloor_{j-1}$$

- by the induction hypothesis it thus follows that

$$\begin{aligned} h' &\in \lfloor \circledast_{z \in T''} q''(z)(l_z) \rfloor_{j-1-k} \\ &= \lfloor q(x)(l_y) * (\circledast_{z \in (T \setminus \{x\})} q(z)(l_z)) * (\circledast_{z \in T'''} \top) \rfloor_{j-i} \\ &= \lfloor q(x)(l_y) * (\circledast_{z \in (T \setminus \{x\})} q(z)(l_z)) \rfloor_{j-i} \end{aligned}$$

  where $l_z = y.l$ if $y \in T'$ and $z.t = y.t$.

- lastly, since $x.t = y.t$, $l_x = l_y$ and thus,

$$h' \in \lfloor (\circledast_{z \in T} q(z)(l_z)) \rfloor_{j-i}$$

□

4.2 Guarded recursion meta-theory

In this section we develop the meta-theory of the guarded recursion in the model. In particular, we define a concrete fixed-point operator on predicates on views, and prove that it defines a fixed-point when applied to guarded definitions (Corollary **??**). We also develop a small theory of guardedness, and prove that the fixed-point operator is non-expansive, even on non-guarded definitions (Lemma 18).

$$\boxed{(-)_{(=)} : \forall \mathcal{X} : \text{ASets}.\ ((\mathcal{X} \to_{ne} \mathit{Prop}) \to_{ne} (\mathcal{X} \to_{ne} \mathit{Prop})) \times \mathbb{N} \to (\mathcal{X} \to_{ne} \mathit{Prop}) \in \text{Sets}}$$

$$\begin{aligned} f_0 &\stackrel{\text{def}}{=} \lambda x \in U(\mathcal{X}).\ \mathbb{N} \times \mathcal{M} \\ f_{i+1} &\stackrel{\text{def}}{=} f(f_i) \end{aligned}$$

Lemma 17.

$$\begin{aligned} &\forall \mathcal{X} \in \mathit{ASets}.\ \forall f, g : (\mathcal{X} \to \mathit{Prop}) \to (\mathcal{X} \to \mathit{Prop}) \in \mathit{ASets}. \\ &\quad \forall i \in \mathbb{N}.\ f =_i g \Rightarrow \forall j \in \mathbb{N}.\ f_j =_i g_j \end{aligned}$$

Proof. By induction on j.

- Case $j = 0$: trivial, as $f_0 = (\lambda x \in U(\mathcal{X}).\ \mathbb{N} \times \mathcal{M}) = g_0$.
- Case $j = k + 1$:
  - by the induction hypothesis, $f_k =_i g_k$
  - hence, by $f =_i g$, and non-expansiveness of g,

$$f_j = f(f_k) =_i g(f_k) =_i g(g_k) = g_j$$

□

Guarded recursion

$$\boxed{\mathit{fix} : \forall \mathcal{X} : \text{ASets}.\ ((\mathcal{X} \to \mathit{Prop}) \to (\mathcal{X} \to \mathit{Prop})) \to (\mathcal{X} \to \mathit{Prop}) \in \text{ASets}}$$

$$\mathit{fix}(f) \stackrel{\text{def}}{=} \lambda x \in U(\mathcal{X}).\ \bigcap_{i \in \mathbb{N}} [f_i(x)]_i$$

where $[-]_i : \mathit{Prop} \to \mathit{Prop} \in \text{ASets}$ is defined as,

$$[p]_i = \{(j, m) \in \mathbb{N} \times \mathcal{M} \mid (j, m) \in p \vee i \leq j\}$$

Lemma 18. *fix is non-expansive.*

$$\forall \mathcal{X} \in \mathit{ASets}.\ \forall f, g : (\mathcal{X} \to \mathit{Prop}) \to (\mathcal{X} \to \mathit{Prop}) \in \mathit{ASets}.$$
$$\forall i \in \mathbb{N}.\ f =_i g \Rightarrow \mathit{fix}(f) =_i \mathit{fix}(g)$$

Proof.

- assume $x \in \mathcal{X}$, $j \in \mathbb{N}$ and $m \in \mathcal{M}$ such that

  $$j < i \qquad\qquad (j, m) \in \mathit{fix}(f)(x) = \bigcap_{i \in \mathbb{N}} [f_i(x)]_i$$

- hence, for every $k \in \mathbb{N}$

  $$(j, m) \in [f_k(x)]_k \Leftrightarrow (j, m) \in f_k(x) \vee k \leq j$$

- hence, for every $k > j$, $(j, m) \in f_k(x)$
- furthermore, by Lemma 17, $f_k =_i g_k$
- hence, for every $k > j$,

  $$(j, m) \in g_k(x)$$

  and thus $(j, m) \in \bigcap_{k \in \mathbb{N}} [g_k(x)]_k = \mathit{fix}(g)(x)$

□

Guarded $\boxed{\mathit{guarded} : \forall \mathcal{X} : \text{ASets}.((\mathcal{X} \to \mathit{Prop}) \to (\mathcal{X} \to \mathit{Prop})) \to \mathit{Spec} \in \text{ASets}}$

$$\mathit{guarded}(f) \stackrel{\text{def}}{=} \{i \in \mathbb{N} \mid \forall j \leq i.\ \forall p, q \in X \to \mathcal{V}.\ p =_j q \Rightarrow f(p) =_{j+1} f(q)\}$$

Lemma 19. *guarded is non-expansive.*

$$\forall \mathcal{X} \in \mathit{ASets}.\ \forall f, g : (\mathcal{X} \to \mathit{Prop}) \to (\mathcal{X} \to \mathit{Prop}) \in \mathit{ASets}.$$
$$\forall i \in \mathbb{N}.\ f =_i g \Rightarrow \mathit{guarded}(f) =_i \mathit{guarded}(g)$$

Proof.

- assume $k \leq j < i$ such that $k \in \mathit{guarded}(f)$
- assume $p, q \in \mathcal{X} \to \mathit{Prop}$ such that $p =_k q$
- since $k \in \mathit{guarded}(f)$, $f(p) =_{k+1} f(q)$
- since $k \leq j < i$, $k + 1 \leq i$ and thus

$$g(p) =_{k+1} f(p) =_{k+1} f(q) =_{k+1} g(q)$$

- hence, $k \in \mathit{guarded}(g)$

□

Lemma 20.

$$\forall \mathcal{X} \in \mathit{ASets}.\ \forall f : (\mathcal{X} \to \mathit{Prop}) \to (\mathcal{X} \to \mathit{Prop}) \in \mathit{ASets}.$$
$$\forall i \in \mathbb{N}.\ i \in \mathit{guarded}(f) \Rightarrow f_i =_i f_{i+1}$$

Proof. By induction on $i \in \mathbb{N}$.

- Case $i = 0$: trivial, as there exists no $j < 0$
- Case $i = j + 1$:
  - by the induction hypothesis, $f_i =_i f_{i+1}$
  - hence, since $i \in \mathit{guarded}(f)$

$$f_{i+1} = f(f_i) =_{i+1} f(f_{i+1}) = f_{i+2}$$

□

Corollary 1.

$$\forall \mathcal{X} \in \mathit{ASets}.\ \forall f : (\mathcal{X} \to \mathit{Prop}) \to (\mathcal{X} \to \mathit{Prop}) \in \mathit{ASets}.$$
$$\forall i \in \mathbb{N}.\ i \in \mathit{guarded}(f) \Rightarrow \forall j \in \mathbb{N}.\ f_i =_i f_{i+j}$$

Proof. By induction on $j \in \mathbb{N}$.

- case $j = 0$: trivial, as $=_i$ is an equivalence relation
- case $j = k + 1$:
  - by the induction hypothesis, $f_i =_i f_{i+k}$
  - hence, as $i \in \mathit{guarded}(f)$,

$$f_i =_i f_{i+1} = f(f_i) =_{i+1} f(f_{i+k}) = f_{i+j}$$

    - and thus, by downwards-closure of $=$, $f_i =_i f_{i+j}$

$\square$

Lemma 21.

$$\begin{aligned}&\forall \mathcal{X} \in \mathit{ASets}.\ \forall f : (\mathcal{X} \to \mathit{Prop}) \to (\mathcal{X} \to \mathit{Prop}) \in \mathit{ASets}.\\ &\quad \forall i \in \mathbb{N}.\ i \in \mathit{guarded}(f) \Rightarrow f_i =_i \mathit{fix}(f)\end{aligned}$$

Proof.

- suppose $x \in \mathcal{X}$, $j < i$, and $m \in \mathcal{M}$, then we need to show that

$$(j, m) \in f_i(x) \Leftrightarrow (j, m) \in \bigcap_{k \in \mathbb{N}} [f_k(x)]_k$$

- assume $(j, m) \in \bigcap_{k \in \mathbb{N}} [f_k(x)]_k$
    - then, in particular,

$$(j, m) \in [f_i(x)]_i = \{(j, m) \mid (j, m) \in f_i(x) \vee i \leq j\}$$

    - and thus $(j, m) \in f_i(x)$, as $j < i$
- assume $(j, m) \in f_i(x)$
    - clearly

$$\bigcap_{k \in \mathbb{N}} [f_k(x)]_k = \left(\bigcap_{k \leq j} [f_k(x)]_k\right) \cap \left(\bigcap_{k > j} [f_k(x)]_k\right)$$

    - and

$$(j, m) \in \bigcap_{k \leq j} [f_k(x)]_k$$

      by definition of $[-]_i$
    - to prove $(j, m) \in \bigcap_{k > j} [f_k(x)]_k$, assume $k \in \mathbb{N}$ such that $j < k$
    - case $i \leq k$:
        * then, by Corollary 1, $f_i =_i f_k$
        * and thus $(j, m) \in f_k(x)$ as $j < i$
    - case $k \leq i$:
        * then, by Corollary 1, $f_k =_k f_i$
        * and thus $(j, m) \in f_k(x)$ as $j < k$

$\square$

Corollary 2.

$$\forall \mathcal{X} \in \mathit{ASets}.\ \forall f : (\mathcal{X} \to \mathit{Prop}) \to (\mathcal{X} \to \mathit{Prop}) \in \mathit{ASets}.$$
$$\forall i \in \mathbb{N}.\ i \in \mathit{guarded}(f) \Rightarrow \mathit{fix}(f) =_i f(\mathit{fix}(f))$$

Proof. By case analysis on $i \in \mathbb{N}$.

- Case $i = 0$: trivial, as there is no $j < 0$
- Case $i = j + 1$:
  - by Lemma 21
    $$\mathit{fix}(f) =_{j+1} f_{j+1}, \qquad\qquad f_j =_j \mathit{fix}(f)$$
  - by guardedness of f we thus have that
    $$\mathit{fix}(f) =_{j+1} f_{j+1} = f(f_j) =_{j+1} f(\mathit{fix}(f))$$

□

Later

$$\boxed{\begin{array}{l} \triangleright : \mathit{Prop} \to \mathit{Prop} \in \text{ASets} \\ \triangleright : \mathit{Spec} \to \mathit{Spec} \in \text{ASets} \end{array}}$$

$$\triangleright p \stackrel{\text{def}}{=} \{(i+1, m) \in \mathbb{N} \times \mathcal{M} \mid (i, m) \in p\} \cup (\{0\} \times \mathcal{M})$$
$$\triangleright s \stackrel{\text{def}}{=} \{i + 1 \in \mathbb{N} \mid i \in s\} \cup \{0\}$$

Lemma 22.

$$\forall i \in \mathbb{N}.\ i \in \mathit{stable}(p) \Rightarrow i + 1 \in \mathit{stable}(\triangleright p)$$

Proof.

- let $j \leq i + 1$, $(m_1, m_2) \in R_j^{\mathit{RType}}$ and $(j, m_1) \in \triangleright p$
- case $j = 0$:
  - then $(j, m_2) \in \triangleright p$ holds trivially
- case $j = k + 1$:
  - then $(k, m_1) \in p$
  - by Lemma 6, $(m_1, m_2) \in R_k^{\mathit{RType}}$
  - since $k \leq i$ it thus follows by stability of p that $(k, m_2) \in p$
  - and thus $(j, m_2) \in \triangleright p$

$\square$

Lemma 23.

$$\forall p, q \in \mathit{Prop}.\ p =_i q \Rightarrow \triangleright p =_{i+1} \triangleright q$$

Lemma 24.

$$\begin{aligned}&\forall \mathcal{X} \in \mathit{ASets}.\ \forall f, g, h \in (\mathcal{X} \to \mathit{Prop}) \to (\mathcal{X} \to \mathit{Prop}).\\ &\quad \mathit{guarded}(g) \subseteq \mathit{guarded}(h \circ g \circ f)\end{aligned}$$

Proof.

- let $p, q \in \mathcal{X} \to \mathit{Prop}$ such that $p =_i q$
- then by non-expansiveness of f, $f(p) =_i f(q)$
- hence, by guardedness of g, $g(f(p)) =_{i+1} g(f(q))$
- and thus, by non-expansiveness of h, $g(h(f(p))) =_{i+1} h(g(f(q)))$

$\square$

Lemma 25.

$$\forall \mathcal{X} \in \mathit{ASets}.\ \mathit{guarded}(\lambda p \in \mathcal{X} \to \mathit{Prop}.\ \lambda x \in \mathcal{X}.\ \triangleright p(x))$$

Proof.

- let $j \leq i$ and $p, q \in \mathcal{X} \to \mathit{Prop}$ such that $p =_j q$
- let $x \in \mathcal{X}$
- then $p(x) =_j q(x)$ and thus, by Lemma 23, $\triangleright p(x) =_{j+1} \triangleright q(x)$

$\square$

Lemma 26.

$$\begin{aligned}&\forall \mathcal{X} \in \mathit{ASets}.\ \forall f : \mathit{Prop} \to \mathit{Prop} \in \mathit{ASets}.\\ &\quad \mathit{guarded}(\lambda p \in \mathcal{X} \to \mathit{Prop}.\ \lambda x \in \mathcal{X}.\ \triangleright f(p(x)))\end{aligned}$$

Proof.

- let $j \leq i$ and $p, q \in \mathcal{X} \to \mathit{Prop}$ such that $p =_j q$
- let $x \in \mathcal{X}$
- then $p(x) =_j q(x)$
- hence, by non-expansiveness of f, $f(p(x)) =_j f(q(x))$
- and thus, by Lemma 23, $\triangleright f(p(x)) =_{j+1} \triangleright f(q(x))$

$\square$

4.3 Embedding meta-theory

In this section we develop the meta-theory related to the embedding of specifications into assertions. The main result is the soundness of the AsnI and AsnE rules from Section 2.3.2 (Lemma 35 and Corollary 3).

The main difficulty is proving the soundness of the AsnI rule (Lemma 35), which allows us to move an embedded specification from the pre-condition into the specification context. Very informally, if

$$\Gamma \mid \mathsf{S} \vdash \{\mathsf{P}\}\bar{\mathsf{s}}\{\mathsf{Q}\}$$

holds, then for every $i \in [\![\mathsf{S}]\!]$, $\bar{\mathsf{s}}$ is safe for i steps relative to $[\![\mathsf{P}]\!]$ and $[\![\mathsf{Q}]\!]$. Hence, if we run $\bar{\mathsf{s}}$ for one step, then for every $i \in [\![\mathsf{S}]\!]$ there exists a stable assertion p'_i describing the intermediate state, such that the continuation is safe for $i-1$ steps relative to p'_i and $[\![\mathsf{Q}]\!]$. To prove $\bar{\mathsf{s}}$ safe for i steps relative to $[\![\mathsf{P} * \mathsf{asn}(\mathsf{S})]\!]$ and $[\![\mathsf{Q}]\!]$ we need a *single* stable assertion to describe the intermediate state. The idea is to take this assertion to be the disjunction of all the p'_i assertions, only with each p'_i cut-off at step-index $i-1$, as the continuation is only safe for $i-1$ steps relative to p'_i. To formalize this we define a cutoff operator, written $\lceil p \rceil^i$, which is false at every step-index $j > i$. Next, we prove that given an $\mathbb{N}$ indexed family of pre-conditions p, such that for any $n \in \mathbb{N}$, s is safe for n steps relative to $p(n)$ and q, then s is safe for any number of steps relative to $\bigvee_n \lceil p(n) \rceil^n$ and q (see Lemma 34).

Specification embedding $\boxed{asn : Spec \to Prop}$

$$asn(X) \stackrel{\text{def}}{=} \{(i,m) \in \mathbb{N} \times \mathcal{M} \mid i \in X\}$$

Validity embedding $\boxed{valid : Prop \to Spec}$

$$valid(p) \stackrel{\text{def}}{=} \{i \in \mathbb{N} \mid \forall m \in \mathcal{M}.\ (i,m) \in p\}$$

View cutoff $\boxed{\lceil - \rceil^{=} : \mathcal{V} \times \mathbb{N} \to \mathcal{V}}$

$$\lceil p \rceil^i \stackrel{\text{def}}{=} \{(j,m) \in \mathbb{N} \times \mathcal{M} \mid (j,m) \in p \wedge j \leq i\}$$

Lemma 27.

$$\forall i \in \mathbb{N}.\ \forall p, q \in \mathcal{V}.\ i \in (p \sqsubseteq q) \Rightarrow \forall j \in \mathbb{N}.\ j \in (\lceil p \rceil^i \sqsubseteq q)$$

Proof.

- let $k \in \mathbb{N}$, $r \in \mathcal{V}$, $m \in \mathcal{M}$ and $h \in Heap$ such that

$$0 \leq k \leq j \qquad k \in stable(r) \qquad (k,m) \in \lceil p \rceil^i * r \qquad h \in \lfloor m \rfloor$$

- then $k \leq i$ and $(k, m) \in p * r$
- hence, there exists an $m' \in \mathcal{M}$ such that $(k, m') \in q * r$ and $h \in \lfloor m' \rfloor$

$\square$

Lemma 28.

$$\forall i \in \mathbb{N}.\ \forall p, q \in \mathcal{V}.\ \lceil p * q \rceil^i = \lceil p \rceil^i * \lceil q \rceil^i$$

Lemma 29.

$$\forall i, j \in \mathbb{N}.\ \forall p \in \mathcal{V}.\ i \in \mathit{stable}(p) \Rightarrow j \in \mathit{stable}(\lceil p \rceil^i)$$

Proof.

- let $k \leq j$, $(m_1, m_2) \in R_k^{\mathit{RType}}$ and $(k, m_1) \in \lceil p \rceil^i$
- then $k \leq i$ and thus by i-stability of p, $(k, m_2) \in p$

$\square$

Lemma 30.

$$\begin{aligned} &\forall i \in \mathbb{N}.\ \forall a \in \mathit{Act}.\ \forall p, q \in \mathcal{V}. \\ &\qquad i \in (a\ \mathit{sat}\ \{p\}\{q\}) \Rightarrow \forall j \in \mathbb{N}.\ j \in (a\ \mathit{sat}\ \{\lceil p \rceil^i\}\{\lceil q \rceil^{i-1}\}) \end{aligned}$$

Proof.

- suppose

$$\begin{gathered} k \in \mathbb{N}, \qquad r \in \mathcal{V}, \qquad 1 \leq k \leq j, \qquad k \in \mathit{stable}(r) \\ m \in \mathcal{M}, \qquad s \in \lfloor m \rfloor, \qquad (k, m) \in \lceil p \rceil^i * r, \qquad s' \in [\![a]\!](s) \end{gathered}$$

- then $k \leq i$ and $(k, m) \in p * r$ and thus $(k, m) \in p * \lceil r \rceil^k$
- by assumption there thus exists an $m' \in \mathcal{M}$ such that

$$(k - 1, m') \in q * \lceil r \rceil^k, \qquad s' \in \lfloor m' \rfloor$$

- hence, $(k - 1, m') \in \lceil q \rceil^{i-1} * r$, as $1 \leq k \leq i$

$\square$

Lemma 31.

$$\forall i \in \mathbb{N}.\ \forall a \in \mathit{Act}.\ \forall p, q \in \mathcal{V}.\ i \in (a\ \mathit{sat}\ \{\lceil p \rceil^0\}\{q\})$$

Lemma 32.

$$\forall p_1, p_2, q_1, q_2 \in \mathcal{V}.\ (p_1 * q_1) \cup (p_2 * q_2) \subseteq (p_1 \cup p_2) * (q_1 \cup q_2)$$

Lemma 33.

$$\begin{array}{l}\forall i \in \mathbb{N}.\ \forall a \in \mathit{Act}.\ \forall p_1, p_2, q_1, q_2 \in \mathcal{V}. \\ \quad (a\ \mathit{sat}\ \{p_1\}\{q_1\}) \cap a\ \mathit{sat}\ \{p_2\}\{q_2\}) \subseteq (a\ \mathit{sat}\ \{p_1 \cup p_2\}\{q_1 \cup q_2\})\end{array}$$

Proof.

- suppose

$$\begin{array}{ccc}1 \leq j \leq i, & r \in \mathcal{V}, & j \in \mathit{stable}(r) \\ (j, m) \in (p_1 \cup p_2) * r, & s \in \lfloor m \rfloor, & s' \in [\![a]\!](s)\end{array}$$

- by definition there exists $m_1, m_2 \in \mathcal{M}$ such that

$$m \in m_1 \bullet m_2, \qquad (j, m_1) \in p_1 \cup p_2, \qquad (j, m_2) \in r$$

- Case $(j, m_1) \in p_1$:
  - since $(j, m) \in p_1 * r$ there exists an $m' \in \mathcal{M}$ such that

$$(j, m') \in q_1 * r, \qquad s' \in \lfloor m' \rfloor$$

  - hence $(j, m') \in (q_1 \cup q_2) * r$
- Case $(j, m_1) \in p_2$: as above

□

Lemma 34.

$$\begin{array}{l}\forall i \in \mathbb{N}.\ \forall x \in \mathit{Thread}.\ \forall p \in \mathbb{N} \to \mathcal{V}.\ \forall q \in \mathit{Stack} \to \mathcal{V}. \\ \quad (\forall j \in \mathbb{N}.\ j \in \mathit{safe}(x, p_j, q)) \Rightarrow i \in \mathit{safe}(x, \bigcup_{j \in \mathbb{N}} \lceil p_j \rceil^j, q)\end{array}$$

Proof. By induction on i. As the base case is trivial, assume $i > 0$.

- Case $\mathit{irr}(x)$:
  - by assumption

$$\forall i \in \mathbb{N}.\ i \in (p_i \sqsubseteq q(x.l))$$

  - hence, by Lemma 27

$$\forall i, j \in \mathbb{N}.\ i \in (\lceil p_j \rceil^j \sqsubseteq q(x.l))$$

  - and thus
$$\forall i \in \mathbb{N}.\ i \in (\bigcup_j \lceil p_j \rceil^j \sqsubseteq q(x.l))$$

- Case $x \xrightarrow{a} \{y\} \uplus T$:
  - by assumption, for each $j \in \mathbb{N}_+$ there exists a
$$p'_j : \{y\} \uplus T \to \mathcal{V}$$
    such that
$$\begin{aligned}
&\forall z \in \{y\} \uplus T.\ j \in \mathit{stable}(p'_j(z)) \\
&j \in (a\ \mathit{sat}\ \{p_j\}\{p'_j(y) * (\circledast_{z \in T} p'_j(z))\}) \\
&\mathit{safe}_{j-1}(y, p'_j(y), q) \\
&\forall z \in T.\ \mathit{safe}_{j-1}(z, p'_j(z), \lambda\_.\ \top)
\end{aligned}$$
  - hence, Lemmas 30 and 28,
$$\forall k \in \mathbb{N}.\ \forall j \in \mathbb{N}_+.\ k \in (a\ \mathit{sat}\ \{\lceil p_j \rceil^j\}\{\lceil p'_j(y) \rceil^{j-1} * (\circledast_{z \in T} \lceil p'_j(z) \rceil^{j-1})\})$$
  - by Lemmas 32 and 33 we thus have that
$$\forall k \in \mathbb{N}.\ k \in (a\ \mathit{sat}\ \{\bigcup_{j \in \mathbb{N}_+} \lceil p_j \rceil^j\}\{\left(\bigcup_{j \in \mathbb{N}_+} \lceil p'_j(y) \rceil^{j-1}\right) * \left(\circledast_{z \in T} \bigcup_{j \in \mathbb{N}_+} \lceil j'_i(z) \rceil^{j-1}\right)\})$$
  - and thus, by Lemmas 31 and 33,
$$\forall k \in \mathbb{N}.\ k \in (a\ \mathit{sat}\ \{\bigcup_{j \in \mathbb{N}} \lceil p_j \rceil^j\}\{\left(\bigcup_{j \in \mathbb{N}_+} \lceil p'_j(y) \rceil^{j-1}\right) * \left(\circledast_{z \in T} \bigcup_{j \in \mathbb{N}_+} \lceil p'_j(z) \rceil^{j-1}\right)\})$$
  - and by the induction hypothesis
$$\begin{aligned}
&\mathit{safe}_{i-1}(y, \bigcup_{j \in \mathbb{N}} \lceil p'_{j+1}(y) \rceil^j, q) \\
&\forall z \in T.\ \mathit{safe}_{i-1}(z, \bigcup_{j \in \mathbb{N}} \lceil p'_{j+1}(z) \rceil^j, \lambda\_.\ \top)
\end{aligned}$$
  - furthermore, by Lemma 29, it follows that,
$$\forall z \in \{y\} \uplus T.\ \forall i \in \mathbb{N}.\ i \in \mathit{stable}(\bigcup_{j \in \mathbb{N}} \lceil p'_{j+1}(z) \rceil^j)$$

- lastly

$$\forall z \in \{y\} \uplus T.\ \bigcup_{j\in\mathbb{N}} \lceil p'_{j+1}(z)\rceil^{j} = \bigcup_{j\in\mathbb{N}_+} \lceil p'_j(z)\rceil^{j-1}$$

$\square$

Lemma 35.

$$\begin{array}{l}\forall p, q \in \mathit{Stack} \to \mathcal{V}.\ \forall X \in \mathcal{P}^{\downarrow}(\mathbb{N}).\ \forall s \in \mathit{seq}\ \mathit{Stm}.\\ \quad X \subseteq \mathit{safe}(s, p, q) \Rightarrow (\forall i \in \mathbb{N}.\ i \in \mathit{safe}(s, \lambda l.\ p(l) * \mathit{asn}(X), q))\end{array}$$

Proof.

- suppose $i \in \mathbb{N}$; if $i = 0$ then s is trivially safe; assume $i > 0$
- let $t \in \mathit{TId}$ and $l \in \mathit{Stack}$
- Case $\mathit{irr}((t, l, s))$:
  - then it suffices to show that $i \in (p(l) * \mathit{asn}(X)) \sqsubseteq q(l)$
    - let $j \in \mathbb{N}$, $r \in \mathcal{V}$, $m \in \mathcal{M}$ and $h \in \mathit{Heap}$ such that

$$0 \le j \le i \qquad j \in \mathit{stable}(r) \qquad (j, m) \in p(l) * \mathit{asn}(X) * r \qquad h \in \lfloor m \rfloor$$

    - then $j \in X$ and thus $j \in (p(l) \sqsubseteq q(l))$
    - hence, there exists an $m' \in \mathcal{M}$ such that $(j, m') \in q(l) * r$ and $h \in \lfloor m' \rfloor$
- Case $(t, l, s) \xrightarrow{a} \{y\} \uplus T$:
  - by assumption, for every $j \in X$ there exists a $p'_j \in \{y\} \uplus T \to \mathcal{V}$ such that

$$\begin{array}{l}\forall z \in \{y\} \uplus T.\ j \in \mathit{stable}(p'_j(z))\\ j \in (a\ \mathit{sat}\ \{p\}\{p'_j(y) * (\circledast_{z\in T} p'_j(z))\})\\ \mathit{safe}_{j-1}(y, p'_j(y), q)\\ \forall z \in T.\ \mathit{safe}_{j-1}(z, p'_j(z), \lambda\_.\ \top)\end{array}$$

  - for each $j \in \mathbb{N}$, take $p''_j \in \{y\} \uplus T \to \mathcal{V}$ to be

$$p''_j = \begin{cases} p'_j & \text{if } j \in X \\ \lambda z.\ \bot & \text{otherwise} \end{cases}$$

  - then for every $j \in \mathbb{N}$

$$\begin{array}{l}\mathit{safe}_j(y, p''_{j+1}(y), q)\\ \forall z \in T.\ \mathit{safe}_j(z, p''_{j+1}(z), \lambda\_.\ \top)\end{array}$$

– hence, by Lemma 34,

$$\begin{aligned}&\mathit{safe}_i(y, \bigcup_j \lceil p''_{j+1}(y) \rceil^j, q)\\&\forall z \in T.\ \mathit{safe}_i(z, \bigcup_j \lceil p''_{j+1}(z) \rceil^j, \lambda\_.\ \top)\end{aligned}$$

– take $p' \in \{y\} \uplus T \to \mathcal{V}$ to be

$$p'(z) = \bigcup_j \lceil p''_{j+1}(z) \rceil^j$$

– then $\forall z \in \{y\} \uplus T.\ \forall i \in \mathbb{N}.\ i \in \mathit{stable}(p'(z))$

– it thus suffices to prove that,

$$i \in (a\ \mathit{sat}\ \{p * \mathit{asn}(X)\}\{p'(y) * (\circledast_{z \in T} p'(z))\})$$

  ∗ suppose $j \in \mathbb{N}$, $1 \leq j \leq i$, $r \in \mathcal{V}$, $m \in \mathcal{M}$ and $h, h' \in \mathit{Heap}$ such that

$$j \in \mathit{stable}(r) \qquad (j, m) \in p * \mathit{asn}(X) * r \qquad h \in \lfloor m \rfloor \qquad h' \in [\![a]\!](h)$$

  ∗ then $j \in X$ and $(j, m) \in p * r$

  ∗ hence $j \in (a\ \mathit{sat}\ \{p\}\{p'_j(y) * (\circledast_{z \in T} p'_j(z))\})$

  ∗ there thus exists an $m' \in \mathcal{M}$ such that $h' \in \lfloor m' \rfloor$ and

$$(j-1, m') \in p'_j(y) * (\circledast_{z \in T} p'_j(z)) * r$$

  ∗ hence

$$(j-1, m') \in p'(y) * (\circledast_{z \in T} p'(z)) * r$$

□

Lemma 36.

$$\begin{aligned}&\forall p \in \mathcal{V}.\ \forall X \in \mathcal{P}^{\downarrow}(\mathbb{N}).\\&\qquad X \subseteq (p \sqsubseteq p * \mathit{asn}(X))\end{aligned}$$

Proof.

- let $i \in X$
- suppose $j \in \mathbb{N}$ such that $0 \leq j \leq i$ and $r \in \mathcal{V}$ such that $j \in \mathit{stable}(r)$
- by downwards-closure of X, $j \in X$
- hence

$$\forall m \in \mathcal{M}.\ (j, m) \in p * r \Rightarrow (j, m) \in p * \mathit{asn}(X) * r$$

- and thus
$$\lfloor p * r \rfloor_j \subseteq \lfloor p * asn(X) * r \rfloor_j$$

$\square$

Corollary 3.

$$\forall p, q \in \mathit{Stack} \to \mathcal{V}.\ \forall X \in \mathcal{P}^{\downarrow}(\mathbb{N}).$$
$$(\forall i \in \mathbb{N}.\ i \in \mathit{safe}(s, p * asn(X), q)) \Rightarrow (X \subseteq \mathit{safe}(s, p, q))$$

4.4 CAP meta-theory

In this section we develop the CAP meta-theory of the model. The main results are the soundness of rule StableClosed (Lemma 56) and OpenA (Lemma 64).

Region assertion $\boxed{\mathit{region} : \Delta(\mathit{RId}) \times \Delta(\mathit{RType}) \times \Delta(\mathit{Val}) \times \mathit{Prop} \to \mathit{Prop} \in \text{ASets}}$

$$\begin{aligned}\mathit{region}(r,t,a,p) \stackrel{\text{def}}{=} \{&(i,(l,s,\varsigma)) \in \mathbb{N} \times \mathcal{M} \mid \\ & r \in \mathit{dom}(s) \wedge s(r).t = t \wedge s(r).a = a \wedge (i,(s(r).l,s,\varsigma)) \in p\}\end{aligned}$$

We use $\boxed{p}^{r,t,a}$ as shorthand for $\mathit{region}(r,t,a,p)$.

Action interpretation $\boxed{\mathit{act} : \mathit{RType} \times (\mathit{Val} \times \mathit{AId} \times \mathit{Val} \to \mathcal{V})^2 \to \mathit{AArg} \to \mathit{Action}}$

The *act* function takes as argument a region type and an action pre- and post-condition and interprets them as an action. Since actions are step-indexed relations on shared states, the interpretation ignores the action model component of the action pre- and post-condition, by existentially quantifying over the action model.

$$\begin{aligned}\mathit{act}(t,p,q) \stackrel{\text{def}}{=} \ & \lambda(a,r,\alpha) \in \mathit{Val} \times \mathit{RId} \times \mathit{AId}. \\ & \{(i,s_1,s_2) \in \mathbb{N} \times \mathit{SState} \times \mathit{SState} \mid \\ & \quad \exists l_1, l_2 \in \mathit{LState}.\ \exists \varsigma_1, \varsigma_2 \in \mathit{AMod}.\ \exists v \in \mathit{Val}. \\ & \quad\quad (i,(l_1,s_1,\varsigma_1)) \in p(a,\alpha,v) \wedge (i,(l_2,s_2,\varsigma_2)) \in q(a,\alpha,v) \wedge \\ & \quad\quad s_1(r) = (l_1,t,a) \wedge s_2(r) = (l_2,t,a) \wedge s_1 \le s_2 \wedge \varsigma_1 \le \varsigma_2\}\end{aligned}$$

Protocol assertion

$$\boxed{\mathit{protocol} : \Delta(\mathit{RType}) \times (\Delta(\mathit{Val}) \times \Delta(\mathit{AId}) \times \Delta(\mathit{Val}) \to \mathit{Prop})^2 \to \mathit{Prop} \in \text{ASets}}$$

The *protocol* function corresponds to the protocol assertion in the logic.

$$\begin{aligned}\mathit{protocol}(t,p,q) \stackrel{\text{def}}{=} \ & \{(i,(l,s,\varsigma)) \in \mathbb{N} \times \mathcal{M} \mid \\ & \quad \forall x \in \mathit{AArg}.\ \varsigma(t)(x)|_i = \mathit{act}(t,p,q)(x)|_i\}\end{aligned}$$

At index i, it asserts that the protocol on the given region type is i-equal to the protocol given by the action pre- and post-conditions p and q. We specifically do not require strict equality, to ensure that assertion is closed under i-equality, as required. In particular, this ensures that,

$$\forall (i, m_1) \in \mathit{protocol}(t, p, q).\ \forall m_2.\ m_1 =_i m_2 \Rightarrow (i, m_2) \in \mathit{protocol}(t, p, q)$$

Action assertion $\boxed{\mathit{action} : \Delta(\mathit{AId}) \times \Delta(\mathit{RId}) \times \Delta(\mathit{Perm}) \to \mathit{Prop} \in \text{ASets}}$

$$\mathit{action}(\alpha, r, p) \stackrel{\text{def}}{=} \{(i, m) \in \mathbb{N} \times \mathcal{M} \mid p \leq (m.l.c)(r, \alpha)\}$$

Shared state equivalence $\boxed{\equiv_{(-)} : \mathcal{P}(\mathit{RType}) \to \mathcal{P}(\mathit{SState} \times \mathit{SState})}$

$$\begin{aligned} s_1 \equiv_A s_2 \stackrel{\text{def}}{=} & \mathit{dom}(s_1) = \mathit{dom}(s_2) \wedge (\forall r \in \mathit{dom}(s_1).\ s_1(r).t = s_2(r).t \wedge s_1(r).a = s_2(r).a) \wedge \\ & (\forall r \in \mathit{dom}(s_1).\ s_1(r).a \in A \Rightarrow s_1(r).l = s_2(r).l) \end{aligned}$$

Action model restriction $\boxed{(-)|_{(=)} : \mathit{AMod} \times \mathcal{P}(\mathit{RType}) \to \mathit{AMod}}$

$$\varsigma|_A(t) \stackrel{\text{def}}{=} \begin{cases} \lambda \alpha \in \mathit{AArg}. & \\ \quad \{(i, s_1, s_2) \in \mathit{SState} \times \mathit{SState} \mid \exists s_1', s_2' \in \mathit{SState}. & \text{if } t \in \mathit{dom}(\varsigma) \\ \quad\quad s_1 \equiv_A s_1' \wedge s_2 \equiv_A s_2' \wedge (i, s_1', s_2') \in \varsigma(t)(\alpha)\} & \\ \text{undef} & \text{otherwise} \end{cases}$$

Action model extension $\boxed{\leq_{(-)} : \mathcal{P}(\mathit{RType}) \to \mathcal{P}(\mathit{AMod} \times \mathit{AMod})}$

$$\begin{aligned} \varsigma_1 \leq_A \varsigma_2 \stackrel{\text{def}}{=} & \forall t \in \mathit{dom}(\varsigma_1).\ \forall \alpha \in \mathit{AArg}.\ \forall (i, s_1, s_2) \in \varsigma_1(t)(\alpha).\ \exists s_1', s_2' \in \mathit{SState}. \\ & t \in \mathit{dom}(\varsigma_2) \wedge s_1 \equiv_A s_1' \wedge s_2 \equiv_A s_2' \wedge (i, s_1', s_2') \in \varsigma_2(t)(\alpha) \end{aligned}$$

Action model equivalence $\boxed{\equiv_{(-)} : \mathcal{P}(\mathit{RType}) \to \mathcal{P}(\mathit{AMod} \times \mathit{AMod})}$

$$\varsigma_1 \equiv_A \varsigma_2 \stackrel{\text{def}}{=} \varsigma_1 \leq_A \varsigma_2 \wedge \varsigma_2 \leq_A \varsigma_1$$

Protocol purity $\boxed{\mathit{pure}_{\mathit{protocol}} : \mathit{Prop} \to \mathit{Spec} \in \text{ASets}}$

$$\mathit{pure}_{\mathit{protocol}}(p) \stackrel{\text{def}}{=} \{i \in \mathbb{N} \mid \forall j \leq i.\ \forall (j, (l, s, \varsigma)) \in p.\ \forall \varsigma' \in \mathit{AMod}.\ (j, (l, s, \varsigma')) \in p\}$$

Permission purity $\boxed{\mathit{pure}_{\mathit{perm}} : \mathit{Prop} \to \mathit{Spec} \in \mathrm{ASets}}$

$$\mathit{pure}_{\mathit{perm}}(p) = \{i \in \mathbb{N} \mid \forall j \leq i.\ \forall (j, ((h, c), s, \varsigma)) \in p.\ \forall c' \in \mathit{Cap}.\ (j, ((h, c'), s, \varsigma)) \in p\}$$

State purity $\boxed{\mathit{pure}_{\mathit{state}} : \mathit{Prop} \to \mathit{Spec} \in \mathrm{ASets}}$

$$\mathit{pure}_{\mathit{state}}(p) \overset{\text{def}}{=} \{i \in \mathbb{N} \mid \forall j \leq i.\ \forall (l, s, \varsigma) \in p.\ \forall l' \in \mathit{LState}.\ \forall s' \in \mathit{SState}.\ (l', s', \varsigma) \in p\}$$

Single-step view-shift $\boxed{\sqsubseteq^{(-)} : \Delta(\mathcal{P}(\mathit{RType})) \to \mathit{Prop} \times \mathit{Prop} \to \mathit{Spec} \in \mathrm{ASets}}$

$$\begin{aligned} p \sqsubseteq^{A} q \overset{\text{def}}{=} \{i \in \mathbb{N} \mid \forall m \in \mathcal{M}.\ \forall j \in \mathbb{N}.\ 0 \leq j \leq i \ \Rightarrow \\ \lfloor p * \{(j, m)\} \rfloor_j \subseteq \lfloor q * \{(j, m') \mid m\ \hat{R}^{A}_{j}\ m'\} \rfloor_j\} \end{aligned}$$

Single-step atomic action

$\boxed{\mathit{sat}_{(-)} : \Delta(\mathcal{P}(\mathit{RType})) \to \Delta(\mathit{Action}) \times \mathit{Prop} \times \mathit{Prop} \to \mathit{Spec} \in \mathrm{ASets}}$

$$\begin{aligned} & a\ \mathrm{sat}^{A}\ \{p\}\{q\} \overset{\text{def}}{=} \\ & \quad \{i \in \mathbb{N} \mid \forall m \in \mathcal{M}.\ \forall j \in \mathbb{N}.\ 1 \leq j \leq i \ \Rightarrow \\ & \qquad [\![a]\!](\lfloor p * \{(j, m)\} \rfloor_j) \subseteq \lfloor q * \{(j-1, m') \mid m\ \hat{R}^{A}_{j-1}\ m'\} \rfloor_{j-1}\} \end{aligned}$$

Lemma 37.

$$\forall A, B \in \mathcal{P}(\mathit{RType}).\ \forall p, q \in \mathcal{V}.\ \forall a \in \mathit{Act}.\ a\ \mathit{sat}^{A}\ \{p\}\{q\} \subseteq a\ \mathit{sat}^{A \cup B}\ \{p\}\{q\}$$

Lemma 38.

$$\forall A \in \mathcal{P}(\mathit{RType}).\ \forall p, q \in \mathcal{V}.\ \forall a \in \mathit{Act}.\ a\ \mathit{sat}^{A}\ \{p\}\{q\} \subseteq a\ \mathit{sat}\ \{p\}\{q\}$$

Lemma 39.

$$\forall A, B \in \mathcal{P}(\mathit{RType}).\ \forall s_1, s_2 \in \mathit{SState}.\ s_1 \equiv_A s_2 \Rightarrow s_1 \equiv_{A \setminus B} s_2$$

Lemma 40.

$$\forall A, B \in \mathcal{P}(\mathit{RType}).\ \forall \varsigma_1, \varsigma_2 \in \mathit{AMod}.\ \varsigma_1 \leq_A \varsigma_2 \Rightarrow \varsigma_1 \leq_{A \setminus B} \varsigma_2$$

Lemma 41.

$$\forall A \in \mathcal{P}(\mathit{RType}).\ \forall \varsigma \in \mathit{AMod}.\ \varsigma \equiv_A \varsigma|_A$$

Lemma 42.

$$\begin{aligned}&\forall A \in \mathcal{P}(\mathit{RType}).\ \forall \varsigma_1, \varsigma_2 \in \mathit{AMod}.\ \forall t \in \mathit{dom}(\varsigma_1).\ \forall \alpha \in \mathit{AArg}.\\ &\quad \varsigma_1 \leq_A \varsigma_2 \Rightarrow \varsigma_1(t)(\alpha) \subseteq \varsigma_2|_A(t)(\alpha)\end{aligned}$$

Lemma 43.

$$\forall A \in \mathcal{P}(\mathit{RType}).\ \forall \varsigma \in \mathit{AMod}.\ \forall t \in \mathit{dom}(\varsigma).\ \forall \alpha \in \mathit{AArg}.\ \varsigma(t)(\alpha) \subseteq \varsigma|_A(t)(\alpha)$$

Lemma 44.

$$\begin{aligned}&\forall i \in \mathbb{N}.\ \forall l, l' \in \mathit{LState}.\ \forall s_1, s_2, s_1', s_2' \in \mathit{SState}.\ \forall \varsigma_1, \varsigma_2 \in \mathit{AMod}.\\ &\quad (s_1 \equiv_A s_1' \wedge s_2 \equiv_A s_2' \wedge \ulcorner(l, s_1)\urcorner = \ulcorner(l', s_1')\urcorner \wedge\\ &\quad (\forall r \in \mathit{dom}(s_1).\ s_1(r) = s_2(r) \Rightarrow s_1'(r) = s_2'(r)))\ \Rightarrow\\ &\qquad (l, s_1, \varsigma_1)\ \hat{R}_i^{\mathit{RType}}\ (l, s_2, \varsigma_2) \Rightarrow (l', s_1', \varsigma_1|_A)\ \hat{R}_i^{\mathit{RType}}\ (l', s_2', \varsigma_2|_A)\end{aligned}$$

Proof.

- by definition of $\hat{R}_i^{\mathit{RType}}$

  $$s_1 \leq s_2 \qquad\qquad \varsigma_1 \leq \varsigma_2$$

  and there exists a $c = \pi_c(\ulcorner(l, s_1)\urcorner)$ such that for all $r \in \mathit{dom}(s_1)$:

  $$s_1(r) = s_2(r) \vee (\exists \alpha \in \mathit{AId}.\ c(r, \alpha) < 1 \wedge (i, s_1, s_2) \in \varsigma_1(s_1(r).t)(s_1(r).a, r, \alpha))$$

- from the definition of $\equiv_A$ and $(-)|_A$ it thus follows that,

  $$s_1' \leq s_2' \qquad\qquad \varsigma_1|_A \leq \varsigma_2|_A$$

- let $r \in \mathit{dom}(s_1') = \mathit{dom}(s_1)$
  - case $s_1(r) = s_2(r)$:
    * by assumption $s_1'(r) = s_2'(r)$
  - case $c(r, \alpha) < 1$, $(i, s_1, s_2) \in \varsigma_1(s_1(r).t)(s_1(r).a, r, \alpha)$:
    * by definition of $(-)|_A$ it thus follows that

      $$(i, s_1', s_2') \in (\varsigma_1|_A)(s_1(r).t)(s_1(r).a, r, \alpha)$$

□

4.4.1 Support

Support assertion $\boxed{\mathit{supp}_{(-)}(=) : \Delta(\mathcal{P}(\mathit{RType})) \times \mathit{Prop} \to \mathit{Spec} \in \text{ASets}}$

$$\begin{aligned}\mathit{supp}_A(p) \overset{\text{def}}{=} \{i \in \mathbb{N} \mid \forall j \leq i.\ \forall (j, (l, s, \varsigma)) \in p.\ \forall s' \in \mathit{SState}.\ \forall \varsigma' \in \mathit{AMod}.\\ s|_A = s'|_A \wedge \varsigma \equiv_A \varsigma' \Rightarrow (j, (l, s', \varsigma')) \in p\}\end{aligned}$$

We use $\mathit{supp}_A(p_1, ..., p_n)$ as shorthand for $\mathit{supp}_A(p_1) \cap \cdots \cap \mathit{supp}_A(p_n)$.

Lemma 45.

$$\forall A \in \mathcal{P}(\mathit{RType}).\ \forall s_1, s_2 \in \mathit{SState}.\ s_1|_A \equiv_A s_2 \Rightarrow s_1|_A = s_2$$

Proof.

- by definition of $\equiv_A$, $\mathit{dom}(s_1|_A) = \mathit{dom}(s_2)$
- and for every $r \in \mathit{dom}(s_1|_A)$, $((s_1|_A)(r)).t \in A$
- and thus, $((s_1|_A)(r)).l = (s_2(r)).l$

□

Lemma 46.

$$\begin{gathered}\forall A \in \mathcal{P}(\mathit{RType}).\ \forall R \in \mathit{Action}.\ \forall i \in \mathbb{N}.\ \forall s_1, s_2 \in \mathit{SState}.\\ s_1 \leq s_2 \wedge (i, s_1|_A, s_2|_A) \in R \Rightarrow (i, s_1, s_2) \in R\end{gathered}$$

Proof. Follows from the assumption that $\mathit{good}(R)$. □

Lemma 47.

$$\forall A, B \in \mathcal{P}(\mathit{RType}).\ \forall p \in \mathcal{V}.\ \mathit{supp}_A(p) \subseteq \mathit{supp}_{A \cup B}(p)$$

Lemma 48.

$$\begin{aligned}&\forall A \in \mathcal{P}(\mathit{RType}).\ \forall r \in \mathit{RId}.\ \forall t \in \mathit{RType}.\ \forall a \in \mathit{Val}.\ \forall p \in \mathcal{V}.\\ &\quad \mathit{supp}_{A \setminus \{t\}}(p) \subseteq \mathit{supp}_{A \cup \{t\}}(\boxed{p}^{r,t,a})\end{aligned}$$

Proof.

- let $l \in \mathit{LState}$, $s, s' \in \mathit{SState}$, and $\varsigma, \varsigma' \in \mathit{AMod}$ such that

$$(j, (l, s, \varsigma)) \in \boxed{p}^{r,t,a} \qquad s|_{A \cup \{t\}} = s'|_{A \cup \{t\}} \qquad \varsigma \equiv_{A \cup \{t\}} \varsigma'$$

- then $s(r).t = t$ and $(j, (s(r).l, s, \varsigma)) \in p$

- and since $s|_{A\setminus\{t\}} = s'|_{A\setminus\{t\}}$ and $\varsigma \equiv_{A\setminus\{t\}} \varsigma'$ it follows that

$$(j, (s(r).l, s', \varsigma')) \in p$$

- and since region r has region type t it follows that $s(r) = s'(r)$ and thus,

$$(j, (l, s', \varsigma')) \in \boxed{p}^{r,t,a}$$

$\square$

Lemma 49.

$$\begin{aligned}&\forall A \in \mathcal{P}(\mathit{RType}).\ \forall i \in \mathbb{N}.\ \forall t \in \mathit{RType}.\ \forall I_p, I_q \in \mathit{Val} \times \mathit{AId} \times \mathit{Val} \to \mathcal{V}.\\ &\quad (\forall x \in \mathit{Val} \times \mathit{AId} \times \mathit{Val}.\ i \in \mathit{supp}_{A\setminus\{t\}}(I_p(x), I_q(x))) \Rightarrow\\ &\qquad i \in \mathit{supp}_{A\cup\{t\}}(\mathit{protocol}(t, I_p, I_q))\end{aligned}$$

Proof.

- let $j \leq i$, $l \in \mathit{LState}$, $s, s' \in \mathit{SState}$, and $\varsigma, \varsigma' \in \mathit{AMod}$ such that

$$(j, (l, s, \varsigma)) \in \mathit{protocol}(t, I_p, I_q) \qquad s|_{A\cup\{t\}} = s'|_{A\cup\{t\}} \qquad \varsigma \equiv_{A\cup\{t\}} \varsigma'$$

- then

$$\forall x \in \mathit{AArg}.\ \varsigma(t)(x)|_j = \mathit{act}(t, I_p, I_q)(x)|_j$$

- it thus suffices to show that $\forall x \in \mathit{AArg}.\ \varsigma(t)(x)|_j = \varsigma'(t)(x)|_j$
- let $(a, r, \alpha) \in \mathit{AArg}$
- assume $(k, s_1, s_2) \in \varsigma'(t)(a, r, \alpha)|_j$:
  - then, by Lemma 42, $(k, s_1, s_2) \in \varsigma|_{A\cup\{t\}}(t)(a, r, \alpha)$
  - hence, there exists $s_1', s_2' \in \mathit{SState}$ such that

$$s_1 \equiv_{A\cup\{t\}} s_1' \qquad s_2 \equiv_{A\cup\{t\}} s_2' \qquad (k, s_1', s_2') \in \varsigma(t)(a, r, \alpha)$$

  - hence, there exists $l_1, l_2 \in \mathit{LState}$, $\varsigma'', \varsigma''' \in \mathit{AMod}$, and $v \in \mathit{Val}$ such that

$$(k, (l_1, s_1', \varsigma'')) \in I_p(a, \alpha, v) \qquad (k, (l_2, s_2', \varsigma''')) \in I_q(a, \alpha, v)$$

    and

$$s_i'(r) = (l_i, t, a) \qquad s_1' \leq s_2' \qquad \varsigma'' \leq \varsigma'''$$

  - since $s_i \equiv_{A\cup\{t\}} s_i'$ it follows that

$$s_i|_{A\setminus\{t\}} = s_i'|_{A\setminus\{t\}} \qquad s_i(r) = (l_i, t, a)$$

  - since $i \in \mathit{supp}_{A\setminus\{t\}}(I_p(a,\alpha,v), I_q(a,\alpha,v))$ it thus follows that

$$(k,(l_1,s_1,\varsigma'')) \in I_p(a,\alpha,v) \qquad (k,(l_2,s_2,\varsigma''')) \in I_q(a,\alpha,v)$$

  - and thus, $(k,s_1,s_2) \in \varsigma(t)(a,r,\alpha)|_j$

- assume $(k,s_1,s_2) \in \varsigma(t)(a,r,\alpha)|_j$:
  - then there exists $l_1,l_2 \in \mathit{LState}$, $\varsigma'',\varsigma''' \in \mathit{AMod}$, and $v \in \mathit{Val}$ such that

$$(k,(l_1,s_1,\varsigma'')) \in I_p(a,\alpha,v) \qquad (k,(l_2,s_2,\varsigma''')) \in I_q(a,\alpha,v)$$

    and

$$s_i(r) = (l_i,t,a) \qquad s_1 \le s_2 \qquad \varsigma'' \le \varsigma'''$$

  - since $i \in \mathit{supp}_{A\setminus\{t\}}(I_p(a,\alpha,v), I_q(a,\alpha,v))$ we thus have that,

$$(k,(l_1,s_1|_{A\setminus\{t\}},\varsigma'')) \in I_p(a,\alpha,v) \qquad (k,(l_2,s_2|_{A\setminus\{t\}},\varsigma''')) \in I_q(a,\alpha,v)$$

  - and thus $(k, s_1|_{A\setminus\{t\}}, s_2|_{A\setminus\{t\}}) \in \varsigma(t)(a,r,\alpha)$
  - hence, by $\varsigma \equiv_{A\cup\{t\}} \varsigma'$, there exists $s_1', s_2' \in \mathit{SState}$ such that

$$s_1|_{A\setminus\{t\}} \equiv_{A\cup\{t\}} s_1' \qquad s_2|_{A\setminus\{t\}} \equiv_{A\cup\{t\}} s_2' \qquad (k,s_1',s_2') \in \varsigma'(t)(\alpha)$$

  - hence, by Lemmas 39 and 45,

$$s_1' = s_1|_{A\setminus\{t\}} \qquad s_2' = s_2|_{A\setminus\{t\}}$$

  - and thus, by Lemma 46,

$$(k,s_1,s_2) \in \varsigma'(t)(a,r,\alpha)$$

□

Lemma 50.

$$\begin{aligned}&\forall A \in \mathcal{P}(\mathit{RType}).\ \forall p_1,p_2 \in \mathcal{V}.\\ &\qquad \mathit{supp}_A(p_1) \cap \mathit{supp}_A(p_2) \subseteq \mathit{supp}_A(p_1 \cap p_2) \cap \mathit{supp}_A(p_1 \cup p_2)\end{aligned}$$

Lemma 51.

$$\forall A \in \mathcal{P}(\mathit{RType}).\ \forall p_1,p_2 \in \mathcal{V}.\ \mathit{supp}_A(p_1) \cap \mathit{supp}_A(p_2) \subseteq \mathit{supp}_A(p_1 * p_2)$$

Proof.

- let $l \in \mathit{LState}$, $s,s' \in \mathit{SState}$ and $\varsigma,\varsigma' \in \mathit{AMod}$ such that

$$(j,(l,s,\varsigma)) \in p_1 * p_2 \qquad s|_A = s'|_A \qquad \varsigma \equiv_A \varsigma'$$

- hence, there exists $l_1, l_2 \in \mathit{LState}$ such that
$$l = l_1 \bullet l_2 \qquad (j, (l_1, s, \varsigma)) \in p_1 \qquad (j, (l_2, s, \varsigma)) \in p_2$$
- hence, $(j, (l_1, s', \varsigma')) \in p_1$ and $(j, (l_2, s', \varsigma')) \in p_2$ and thus $(j, (l, s', \varsigma')) \in p_1 * p_2$

$\square$

Lemma 52.

$$\forall A \in \mathcal{P}(\mathit{RType}).\ \forall p_1, p_2 \in \mathcal{V}.\ \mathit{supp}_A(p_1) \cap \mathit{supp}_A(p_2) \subseteq \mathit{supp}_A(p_1 \Rightarrow p_2)$$

Proof.

- let $l \in \mathit{LState}$, $s, s' \in \mathit{SState}$ and $\varsigma, \varsigma' \in \mathit{AMod}$ such that
$$j \leq i \qquad (j, (l, s, \varsigma)) \in p_1 \Rightarrow p_2 \qquad s|_A = s'|_A \qquad \varsigma \equiv_A \varsigma'$$
- hence,
$$\forall k \leq j.\ \forall m \geq (l, s, \varsigma).\ (k, m) \in p_1 \Rightarrow (k, m) \in p_2$$
- let $k \leq j$, $(l', s'', \varsigma'') \geq (l, s', \varsigma')$ such that $(k, (l', s'', \varsigma'')) \in p_1$
- define s''' and ς''' as follows,
$$s'''(r) \overset{\text{def}}{=} \begin{cases} s(r) & \text{if } r \in \mathit{dom}(s) \text{ and } s(r).t \notin A \\ s''(r) & \text{otherwise} \end{cases}$$
$$\varsigma'''(r) \overset{\text{def}}{=} \begin{cases} \varsigma(r) & \text{if } r \in \mathit{dom}(\varsigma) \\ \varsigma''(r) & \text{otherwise} \end{cases}$$
- then $(l, s, \varsigma) \leq (l', s''', \varsigma''')$, $s'''|_A = s''|_A$, and $\varsigma'' \equiv_A \varsigma'''$
- hence, $(k, (l', s''', \varsigma''')) \in p_1$, and thus $(k, (l', s''', \varsigma''')) \in p_2$, and thus
$$(k, (l', s'', \varsigma'')) \in p_2$$

$\square$

4.4.2 Stability

Interference decomposition $\boxed{\hat{R}^{(=)}_{(-)} \in \mathbb{N} \times (\mathit{RId} \times \mathcal{P}(\mathit{AId})) \to \mathcal{P}(\mathcal{M} \times \mathcal{M})}$

$$\begin{aligned}
&(l_1, s_1, \varsigma_1)\ \hat{R}^{r,A}_i\ (l_2, s_2, \varsigma_2) \quad \text{iff} \\
&\quad l_1 = l_2 \wedge s_1 \leq s_2 \wedge \varsigma_1 \leq \varsigma_2 \wedge \\
&\qquad \exists c \in \mathit{Cap}.\ c = \pi_c(\ulcorner (l_1, s_1) \urcorner). \\
&\qquad\quad (\forall r' \in \mathit{dom}(s_1).\ s_1(r') = s_2(r') \vee \\
&\qquad\qquad (\exists \alpha \in \mathit{AId}.\ (r' = r \Rightarrow \alpha \in A) \wedge c(r', \alpha) < 1 \\
&\qquad\qquad\quad \wedge\ (i, s_1, s_2) \in \varsigma_1(s_1(r).t)(s_1(r).a, r, \alpha)))
\end{aligned}$$

Stability decomposition $\boxed{\mathit{stable}^{(-)} : \Delta(\mathcal{P}(\mathit{AId}) \times \mathit{RId}) \to \mathit{Prop} \to \mathit{Spec} \in \text{ASets}}$

$$\mathit{stable}^{r,A}(p) \stackrel{\text{def}}{=} \{i \in \mathbb{N} \mid \forall j \leq i.\ \forall m, m' \in \mathcal{M}.\ (j, m) \in p \wedge m\ \hat{R}_i^{r,A}\ m' \Rightarrow (j, m') \in p\}$$

Lemma 53.

$$\forall i \in \mathbb{N}.\ \forall r \in \mathit{RId}.\ \hat{R}_i^{r,\mathit{AId}} = \hat{R}_i^{\mathit{RType}}$$

Lemma 54.

$$\forall i \in \mathbb{N}.\ \forall A_1, A_2 \in \mathcal{P}(\mathit{AId}).\ \forall r \in \mathit{RId}.\ A_1 \subseteq A_2 \Rightarrow \hat{R}_i^{r,A_1} \subseteq \hat{R}_i^{r,A_2}$$

Lemma 55.

$$\begin{array}{l}\forall A_1, A_2 \in \mathcal{P}(\mathit{AId}).\ \forall r \in \mathit{RId}.\ \forall p \in \mathcal{V}. \\ \qquad \mathit{stable}^{r,A_1}(p) \cap \mathit{stable}^{r,A_2}(p) \subseteq \mathit{stable}^{r,A_1 \cup A_2}(p)\end{array}$$

Lemma 56.

$$\begin{array}{l}\forall i \in \mathbb{N}.\ \forall p, q \in \mathcal{V}.\ \forall I_p, I_q : \mathit{Val} \times \mathit{AId} \times \mathit{Val} \to \mathcal{V}. \\ \forall r \in \mathit{RId}.\ \forall t \in \mathit{RType}.\ \forall a \in \mathit{Val}.\ \forall \alpha \in \mathit{AId}. \\ \quad i \in \mathit{stable}(p * q) \wedge i \in \mathit{supp}_{A \setminus \{t\}}(p, q)\ \wedge \\ \quad i \in \mathit{pure}_{\mathit{protocol}}(p) \wedge \mathit{pure}_{\mathit{state}}(q)\ \wedge \\ \quad (\forall v : \mathit{Val}.\ ((I_p(a, \alpha, v) \cap p) \subseteq \bot) \vee (I_q(a, \alpha, v) \subseteq p)) \\ \quad \Rightarrow i \in \mathit{stable}^{r,\{\alpha\}}(\boxed{p * q}_{I_p, I_q}^{r,t,a})\end{array}$$

Proof.

- assume

$$j \leq i \qquad (j, (l, s, \varsigma)) \in \boxed{p * q}_{I_p, I_q}^{r,t,a} \qquad (l, s, \varsigma)\ \hat{R}_j^{r,\{\alpha\}}\ (l, s', \varsigma')$$

- then $(j, (s(r).l, s, \varsigma)) \in p$ and $(j, (l, s, \varsigma)) \in q$
- case $s(r) = s'(r)$:
  - by Lemma 41, $\varsigma \equiv_{A \setminus \{t\}} \varsigma|_{A \setminus \{t\}}$, and hence, since $i \in \mathit{supp}_{A \setminus \{t\}}(p, q)$,

$$(j, (s(r).t, s[r \mapsto (l, t, a)], \varsigma|_{A \setminus \{t\}})) \in p * q$$

  - by Lemma 54 it follows that $(l, s, \varsigma)\ \hat{R}_j^{r,\mathit{AId}}\ (l, s', \varsigma')$ and thus, by Lemma 53,

$$(l, s, \varsigma)\ \hat{R}_j^{\mathit{RType}}\ (l, s', \varsigma')$$

– and since $\ulcorner(l, s)\urcorner = \ulcorner(s(r).l, s[r \mapsto (l, t, a)])\urcorner$,

$$s \equiv_{A\setminus\{t\}} s[r \mapsto (l, t, a)] \qquad\qquad s' \equiv_{A\setminus\{t\}} s'[r \mapsto (l, t, a)]$$

and

$$\begin{array}{l}\forall r' \in dom(s[r \mapsto (l, t, a)]).\\ \qquad s(r') = s'(r') \Rightarrow (s[r \mapsto (l, t, a)])(r') = (s'[r \mapsto (l, t, a)])(r')\end{array}$$

it follows from Lemma 44 that,

$$(s(r).l, s[r \mapsto (l, t, a)], \varsigma|_{A\setminus\{t\}})\ \hat{R}_j^{RType}\ (s(r).l, s'[r \mapsto (l, t, a)], \varsigma'|_{A\setminus\{t\}})$$

– hence, by stability of $p * q$,

$$(j, (s(r).l, s'[r \mapsto (l, t, a)], \varsigma'|_{A\setminus\{t\}})) \in p * q$$

– and, since $i \in supp_{A\setminus\{t\}}(p, q)$,

$$(j, (s'(r).l, s', \varsigma')) \in p * q$$

– hence,

$$(j, (l, s', \varsigma')) \in \boxed{p * q}_{I_p, I_q}^{r,t,a}$$

- case $s(r) \neq s'(r)$:
  – then $(j, s, s') \in \varsigma(t)(a, r, \alpha)$
  – hence, there exists $l_1, l_2 \in LState$, $\varsigma'', \varsigma''' \in AMod$, and $v \in Val$ such that

$$\begin{array}{ll}(j, (l_1, s, \varsigma'')) \in I_p(a, \alpha, v) & s(r) = (l_1, t, a)\\ (j, (l_2, s', \varsigma''')) \in I_q(a, \alpha, v) & s'(r) = (l_2, t, a)\end{array}$$

  – case $I_p(a, \alpha, v) \cap p \subseteq \bot$:
    * since $i \in pure_{protocol}(p)$, $(j, (s(r).l, s, \varsigma'')) \in p$
    * hence $(j, (s(r).l, s, \varsigma)) \in \bot$, which is a contradiction
  – case $I_q(a, \alpha, v) \subseteq p$:
    * then $(j, (s'(r).l, s', \varsigma''')) \in p$
    * since $i \in pure_{protocol}(p)$ we thus have that $(j, (s'(r).l, s', \varsigma')) \in p$
    * furthermore, since $(j, (l, s, \varsigma)) \in q$, $\varsigma \leq \varsigma'$ and $i \in pure_{state}(q)$,

$$(j, (\varepsilon, s', \varsigma')) \in q$$

    and thus $(j, (l, s', \varsigma')) \in \boxed{p * q}_{I_p, I_q}^{r,t,a}$

□

4.4.3 View shifts

Lemma 57.

$$\begin{aligned}&\forall i \in \mathbb{N}.\ \forall A \in \mathcal{P}(\mathit{RType}).\ \forall m_1, m_2, m_3, m_4 \in \mathcal{M}.\\ &\qquad m_1\ \hat{R}_i^{\emptyset}\ m_2 \wedge m_2\ \hat{R}_i^{A}\ m_3 \wedge m_3\ \hat{R}_i^{\emptyset}\ m_4 \Rightarrow m_1\ \hat{R}_i^{A}\ m_4\end{aligned}$$

Lemma 58.

$$\begin{aligned}&\forall A \in \mathcal{P}(\mathit{RType}).\ \forall p, p', q, q' \in \mathit{Prop}.\\ &\quad (p \sqsubseteq^{A} p') \cap (a\ \mathit{sat}^{\emptyset}\ \{p'\}\{q'\}) \cap (q' \sqsubseteq^{\emptyset} q) \subseteq (a\ \mathit{sat}^{A}\ \{p\}\{q\}) \wedge\\ &\quad (p \sqsubseteq^{\emptyset} p') \cap (a\ \mathit{sat}^{A}\ \{p'\}\{q'\}) \cap (q' \sqsubseteq^{\emptyset} q) \subseteq (a\ \mathit{sat}^{A}\ \{p\}\{q\}) \wedge\\ &\quad (p \sqsubseteq^{\emptyset} p') \cap (a\ \mathit{sat}^{\emptyset}\ \{p'\}\{q'\}) \cap (q' \sqsubseteq^{A} q) \subseteq (a\ \mathit{sat}^{A}\ \{p\}\{q\})\end{aligned}$$

Proof.

- let $1 \le j \le i$, $m_1, m_2 \in \mathcal{M}$ and $h, h' \in \mathit{Heap}$ such that

$$(j, m_1) \in p \qquad h \in \lfloor m_1 \bullet m_2 \rfloor \qquad h' \in [\![a]\!](h)$$

- then there exists $m_1', m_2' \in \mathcal{M}$ such that

$$(j, m_1') \in p' \qquad h \in \lfloor m_1' \bullet m_2' \rfloor \qquad m_2\ \hat{R}_j^{A}\ m_2'$$

- hence, there exists $m_1'', m_2'' \in \mathcal{M}$ such that

$$(j-1, m_1'') \in q' \qquad h' \in \lfloor m_1'' \bullet m_2'' \rfloor \qquad m_2'\ \hat{R}_{j-1}^{\emptyset}\ m_2''$$

- hence, there exists $m_1''', m_2''' \in \mathcal{M}$ such that

$$(j-1, m_1''') \in q \qquad h' \in \lfloor m_1''' \bullet m_2''' \rfloor \qquad m_2''\ \hat{R}_{j-1}^{\emptyset}\ m_2'''$$

- hence, by Lemma 57, $m_2\ \hat{R}_{j-1}^{A}\ m_2'''$ and thus

$$(p \sqsubseteq^{A} p') \cap (a\ \mathrm{sat}^{\emptyset}\ \{p'\}\{q'\}) \cap (q' \sqsubseteq^{\emptyset} q) \subseteq (a\ \mathrm{sat}^{A}\ \{p\}\{q\})$$

- the other cases follow the same structure

□

Lemma 59.

$$\begin{aligned}&\forall i \in \mathbb{N}.\ \forall I_p, I_q \in \mathit{Val} \times \mathit{AId} \times \mathit{Val} \to \mathit{Prop}.\ \forall t \in \mathit{RType}.\\ &\qquad i \in (\mathit{emp} \sqsubseteq^{\emptyset} \exists s \in \mathit{RType}.\ t \le s * \mathit{protocol}(s, I_p, I_q))\end{aligned}$$

Proof.

- let $j \leq i$, $l_1, l_2 \in \mathit{LState}$, $s \in \mathit{SState}$, $\varsigma \in \mathit{AMod}$ and $h \in \mathit{Heap}$ such that
$$(j, (l_1, s, \varsigma)) \in \mathit{emp} \qquad\qquad h \in \lfloor (l_1, s, \varsigma) \bullet (l_2, s, \varsigma) \rfloor$$
- by assumption $\mathit{dom}(\varsigma)$ is finite and thus $t^* \setminus \mathit{dom}(\varsigma)$ is infinite
- pick $s \in \mathit{RType}$ such that $s \in t^* \setminus \mathit{dom}(\varsigma)$
- then
$$(l_2, s, \varsigma)\ \hat{R}^{\emptyset}_j\ (l_2, s, \varsigma[s \mapsto \mathit{act}(s, I_p, I_q)])$$
- and
$$(j, (l_1, s, \varsigma[s \mapsto \mathit{act}(s, I_p, I_q)])) \in \mathit{protocol}(s, I_p, I_q)$$
- and lastly,
$$h \in \lfloor (l_1, s, a[s \mapsto \mathit{act}(s, I_p, I_q)]) \bullet (l_2, s, a[s \mapsto \mathit{act}(s, I_p, I_q)]) \rfloor$$

□

Lemma 60.

$$\begin{array}{l}\forall i \in \mathbb{N}.\ \forall A \in \mathcal{P}_{fin}(\mathit{AId}).\ \forall t \in \mathit{RType}.\ \forall a \in \mathit{Val}.\ \forall p \in \mathit{Prop}.\\ \quad i \in (p \sqsubseteq^{\emptyset} \exists r : \mathit{RId}.\ \boxed{p}^{r,t,a} * \circledast_{\alpha \in A} [\alpha]^r_1)\end{array}$$

Proof.

- let $j \leq i$, $l_1, l_2 \in \mathit{LState}$, $s \in \mathit{SState}$, $\varsigma \in \mathit{AMod}$ and $h \in \mathit{Heap}$ such that
$$(j, (l_1, s, \varsigma)) \in p \qquad\qquad h \in \lfloor (l_1, s, \varsigma) \bullet (l_2, s, \varsigma) \rfloor$$
- by assumption $\mathit{dom}(s) \cup \mathit{dom}_r(\pi_c(l_1 \bullet l_2))$ is finite
- pick $r \in \mathit{RId}$ such that $r \notin \mathit{dom}(s) \cup \mathit{dom}_r(\pi_c(l_1 \bullet l_2))$
- then
$$(l_2, s, \varsigma)\ \hat{R}^{\emptyset}_j\ (l_2, s[r \mapsto (l_1, t, a)], \varsigma)$$
- and
$$(j, ((\varepsilon, [(r, A) \mapsto 1]), s[r \mapsto (l_1, t, a)], \varsigma)) \in \boxed{p}^{r,t,a} * \circledast_{\alpha \in A} [\alpha]^r_1$$
- and lastly,
$$h \in \lfloor ((\varepsilon, [(r, A) \mapsto 1]), s[r \mapsto (l_1, t, a)], \varsigma) \bullet (l_2, s[r \mapsto (l_1, t, a)], \varsigma) \rfloor$$

□

Phantom points-to $\boxed{(-)_{(=)} \mapsto (\equiv) : \Delta(\mathit{OId}) \times \Delta(\mathit{FName}) \times \Delta(\mathit{Val}) \to \mathit{Prop} \in \text{ASets}}$

$$x_f \mapsto y \overset{\text{def}}{=} \{(i,m) \in \mathbb{N} \times \mathcal{M} \mid (m.l.p)(x,f) = y\}$$

Lemma 61.

$$\begin{aligned}&\forall t \in \mathit{RType}.\ \forall x \in \mathit{OId}.\ \forall f \in \mathit{FName}.\ \forall v_1, v_2 \in \mathit{Val}.\\ &\qquad \forall i \in \mathbb{N}.\ i \in (x_f \mapsto v_1 \sqsubseteq^{\emptyset} x_f \mapsto v_2)\end{aligned}$$

Proof.

- let $j \leq i$, $r \in \mathit{Prop}$, $m_1, m_2 \in \mathcal{M}$ and $h \in \mathit{Heap}$ such that

$$(j, m_1) \in x_f \mapsto v_1 \qquad\qquad h \in \lfloor m_1 \bullet m_2 \rfloor$$

- hence, $(x,f) \in \mathit{dom}(m_1.l.p)$ and $(x,f) \notin \mathit{dom}(m_2.l.p)$
- thus,

$$(j, m_1[(x,f) \mapsto_p v_2]) \in x_f \mapsto v_2 \qquad \lfloor m_1 \bullet m_2 \rfloor = \lfloor m_1[(x,f) \mapsto_p v_2] \bullet m_2 \rfloor$$

  where $m[(x,f) \mapsto_p v]$ is notation for $((m.l.h, m.l.p[(x,f) \mapsto v], m.l.c), m.s, m.a)$

□

Atomic update allowed

$$\boxed{\leadsto^{(-)}_{(=)} : \Delta(\mathit{RId} \times \mathcal{P}(\mathit{RType})) \times \Delta(\mathit{Action}) \to \mathit{Prop} \times \mathit{Prop} \to \mathit{Spec} \in \text{ASets}}$$

$$\begin{aligned}p \leadsto^{r,A}_{a} q \overset{\text{def}}{=} \{i \in \mathbb{N} \mid\ & \forall j \leq i.\ \forall m_1, m_2 \in \mathcal{M}.\ \forall h_1 \in \lfloor m_1 \rfloor.\ \forall h_2 \in \lfloor m_2 \rfloor.\\ & 1 \leq j \wedge (j, m_1) \in p \wedge (j-1, m_2) \in q \wedge h_2 \in [\![a]\!](h_1) \ \Rightarrow\\ & \quad \exists \alpha \in \mathit{AId}.\\ & \qquad (m_1.l.c)(r, \alpha) > 0\ \wedge\\ & \qquad (j-1, (m_1.s)|_A, (m_2.s)|_A) \in (m_1.a)((m_1.s)(r).t)((m_1.s)(r).a, r, \alpha)\}\end{aligned}$$

View shift allowed $\boxed{\leadsto^{(-)} : \Delta(\mathit{RId} \times \mathcal{P}(\mathit{RType})) \to \mathit{Prop} \times \mathit{Prop} \to \mathit{Spec} \in \text{ASets}}$

$$\begin{aligned}p \leadsto^{r,A} q \overset{\text{def}}{=} \{i \in \mathbb{N} \mid\ & \forall j \leq i.\ \forall m_1, m_2 \in \mathcal{M}.\ \forall h \in \lfloor m_1 \rfloor.\ \forall h \in \lfloor m_2 \rfloor.\\ & (j, m_1) \in p \wedge (j, m_2) \in q \ \Rightarrow\\ & \quad \exists \alpha \in \mathit{AId}.\\ & \qquad (m_1.l.c)(r, \alpha) > 0\ \wedge\\ & \qquad (j, (m_1.s)|_A, (m_2.s)|_A) \in (m_1.a)((m_1.s)(r).t)((m_1.s)(r).a, r, \alpha)\}\end{aligned}$$

Lemma 62.

$\forall i \in \mathbb{N}.\ \forall A \in \mathcal{P}(\mathit{RType}).\ \forall r \in \mathit{RId}.\ \forall t \in \mathit{RType}.\ \forall b \in \mathit{Val}.\ \forall p_1, p_2, q_1, q_2 \in \mathit{Prop}.$

$$\begin{aligned}&\mathit{valid}(p_1 * p_2 \Rightarrow q_1 * q_2) \cap \mathit{supp}_{A\setminus\{t\}}(p_1, p_2, q_1, q_2) \cap (\boxed{p_1}^{r,t,b} * p_2 \rightsquigarrow^{r,A\cup\{t\}} \boxed{q_1}^{r,t,b} * q_2)\\ &\quad\subseteq\ (\boxed{p_1}^{r,t,b} * p_2 \sqsubseteq^{A\cup\{t\}} \boxed{q_1}^{r,t,b} * q_2)\end{aligned}$$

Proof.

- let $j \leq i$, $(l_1, s, \varsigma), (l_2, s, \varsigma), (l_3, s, \varsigma) \in \mathcal{M}$ and $h \in \mathit{Heap}$ such that

$$(j, (l_1, s, \varsigma)) \in \boxed{p_1}^{r,t,b} \qquad\qquad (j, (l_2, s, \varsigma)) \in p_2$$

  and

$$h \in \lfloor (l_1, s, \varsigma) \bullet (l_2, s, \varsigma) \bullet (l_3, s, \varsigma) \rfloor$$

- thus

$$(l_r, s, \varsigma) \in p_1 \qquad\qquad s(r) = (l_1, t, b)$$

  where $l_r = s(r).l$

- since $i \in \mathit{supp}_{A\setminus\{t\}}(p_1, p_2)$, we thus have that

$$(j, (l_r, s_r, \varsigma)) \in p_1 \qquad\qquad (j, (l_2, s_r, \varsigma)) \in p_2$$

  where $s_r = s[r \mapsto (\varepsilon, t, b)]$

- and by upwards-closure, $(j, (l_r \bullet l_1, s_r, \varsigma)) \in p_1$
- hence, there exists $(l'_1, s', \varsigma'), (l'_2, s', \varsigma') \in \mathcal{M}$ such that

$$(j, (l'_1, s', \varsigma')) \in q_1 \qquad\qquad (j, (l'_2, s', \varsigma')) \in q_2$$

  and

$$(l_r \bullet l_1, s_r, \varsigma) \bullet (l_2, s_r, \varsigma) = (l'_1, s', \varsigma') \bullet (l'_2, s', \varsigma')$$

- hence, $s' = s_r$, $\varsigma' = \varsigma$ and $l_r \bullet l_1 \bullet l_2 = l'_1 \bullet l'_2$
- since $i \in \mathit{supp}_{A\setminus\{t\}}(q_1, q_2)$ it follows that

$$(j, (\varepsilon, s'_r, \varsigma)) \in \boxed{q_1}^{r,t,b} \qquad\qquad (j, (l'_2, s'_r, \varsigma)) \in q_2$$

  where $s'_r = s[r \mapsto (l'_1, t, b)]$

- furthermore, as

$$\begin{aligned}\lfloor (l_1, s, \varsigma) \bullet (l_2, s, \varsigma) \bullet (l_3, s, \varsigma) \rfloor &= \lfloor (l_r \bullet l_1, s_r, \varsigma) \bullet (l_2, s_r, \varsigma) \bullet (l_3, s_r, \varsigma) \rfloor \\ &= \lfloor (l_1', s_r, \varsigma) \bullet (l_2', s_r, \varsigma) \bullet (l_3, s_r, \varsigma) \rfloor \\ &= \lfloor (\varepsilon, s_r', \varsigma) \bullet (l_2', s_r', \varsigma) \bullet (l_3, s_r', \varsigma) \rfloor\end{aligned}$$

  it suffices to prove that $(l_3, s, \varsigma)\ \hat{R}_j^{A \cup \{t\}}\ (l_3, s_r', \varsigma)$

- since $i \in (\boxed{p_1}^{r,t,b} * p_2 \rightsquigarrow^{r, A\cup\{t\}} \boxed{q_1}^{r,t,b} * q_2)$ there exists an $\alpha \in \mathit{AId}$ such that

$$\pi_c(l_1 \bullet l_2)(r, \alpha) > 0 \qquad\qquad (j, s|_{A\cup\{t\}}, s_r'|_{A\cup\{t\}}) \in \varsigma(t)(b, r, \alpha)$$

- and for all $r' \in \mathit{dom}(s)$ such that $r \neq r'$, $s(r') = s_r'(r')$

$\square$

Lemma 63.

$\forall i \in \mathbb{N}.\ \forall A, B \in \mathcal{P}(\mathit{RType}).\ \forall l \in \mathit{LState}.\ \forall s_1, s_2, s_1', s_2' \in \mathit{SState}.\ \forall \varsigma_1, \varsigma_2 \in \mathit{AMod}.$
$s_1 \equiv_A s_1' \wedge s_2 \equiv_A s_2' \wedge (\forall r \in \mathit{dom}(s_1'|_{\mathit{RType}\setminus(A\cup B)}).\ s_1'(r) = s_2'(r)) \wedge \ulcorner(l, s_1)\urcorner = \ulcorner(l, s_1')\urcorner \wedge$
$(\forall r \in \mathit{dom}(s_1'|_B).\ \exists \alpha \in \mathit{AId}.\ \pi_c(\ulcorner(l_1, s_1)\urcorner)(r, \alpha) < 1 \wedge$
$(i, s_1'|_{A\cup B}, s_2'|_{A\cup B}) \in \varsigma_1(s_1'(r).t)(s_1'(r).a, r, \alpha)) \Rightarrow$
$(l, s_1, \varsigma_1)\ \hat{R}_i^A\ (l, s_2, \varsigma_2) \Rightarrow (l, s_1', \varsigma_1)\ \hat{R}_i^{A\cup B}\ (l, s_2', \varsigma_2)$

Proof.

- by assumption, $s_1 \leq s_2$ and $\varsigma_1 \leq \varsigma_2$
- since, $s_1 \equiv_A s_1'$ and $s_2 \equiv_A s_2'$, $s_1' \leq s_2'$
- let $r \in \mathit{dom}(s_1') = \mathit{dom}(s_1)$
- case $s_1'(r) = s_2'(r)$: trivial
- case $s_1'(r) \neq s_2'(r)$:
  - case $r \in \mathit{dom}(s_1'|_B)$:
    - by assumption, there exists an $\alpha \in \mathit{AId}$ such that

$$\pi_c(\ulcorner(l, s_1')\urcorner)(r, \alpha) < 1 \quad (i, s_1'|_{A\cup B}, s_2'|_{A\cup B}) \in \varsigma_1(s_1'(r).t)(s_1'(r).a, r, \alpha)$$

  - case $r \in \mathit{dom}(s_1'|_A)$:
    - then $s_1(r) = s_1'(r) \neq s_2'(r) = s_2(r)$
    - there thus exists an $\alpha \in \mathit{AId}$ such that $\pi_c(\ulcorner(l, s_1)\urcorner)(r, \alpha) < 1$ and

$$(i, s_1|_A, s_2|_A) \in \varsigma_1(s_1(r).g)(s_1(r).a, r, \alpha)$$

    * hence, $\pi_c(\ulcorner(l, s'_1)\urcorner)(r, \alpha) < 1$ and

$$(i, s'_1|_{A \cup B}, s'_2|_{A \cup B}) \in \varsigma_1(s'_1(r).t)(s'_1(r).a, r, \alpha)$$

  - case $r \in dom(s'_1|_{RType \setminus (A \cup B)})$: trivial

$\square$

Lemma 64.

$\forall A \in \mathcal{P}(RType).\ \forall a \in Act.\ \forall r \in RId.\ \forall t \in RType.\ \forall b \in Val.\ \forall p_1, p_2, q_1, q_2 \in X \to Prop.$
$supp_{A \setminus \{t\}}(p_1, p_2, q_1, q_2) \cap pure_{perm}(p_1) \cap$
$(\exists x : X.\ \boxed{p_1(x)}^{r,t,b} * p_2(x) \rightsquigarrow_a^{r, A \cup \{t\}} \exists x : X.\ \boxed{q_1(x)}^{r,t,b} * q_2(x)) \cap$
$(a\ sat^{A \setminus \{t\}}\ \{\exists x : X.\ p_1(x) * p_2(x)\}\{\exists x : X.\ q_1(x) * q_2(x)\})$
$\subseteq\ (a\ sat^{A \cup \{t\}}\ \{\exists x : X.\ \boxed{p_1(x)}^{r,t,b} * p_2(x)\}\{\exists x : X.\ \boxed{q_1(x)}^{r,t,b} * q_2(x)\})$

Proof.

- let $1 \leq j \leq i$, $(l_1, s, \varsigma), (l_2, s, \varsigma), (l_3, s, \varsigma) \in \mathcal{M}$, $x \in X$ and $h, h' \in Heap$ such that

$$(j, (l_1, s, \varsigma)) \in \boxed{p_1(x)}^{r,t,b} \qquad\qquad (j, (l_2, s, \varsigma)) \in p_2(x)$$

  and

$$h \in \lfloor (l_1, s, \varsigma) \bullet (l_2, s, \varsigma) \bullet (l_3, s, \varsigma) \rfloor \qquad\qquad h' \in [\![a]\!](h)$$

- thus

$$(l_r, s, \varsigma) \in p_1(x) \qquad\qquad s(r) = (l_1, t, b)$$

  where $l_r = (s(r).l$

- since $i \in supp_{A \setminus \{t\}}(p_1, p_2) \cap pure_{perm}(p_1)$, we thus have that

$$(j, ((l_r.h, l_r.p, \varepsilon), s_r, \varsigma)) \in p_1(x) \qquad\qquad (j, (l_2, s_r, \varsigma)) \in p_2(x)$$

  where $s_r = s[r \mapsto ((\varepsilon, \varepsilon, l_r.c), t, b)]$

- and by upwards-closure, $(j, ((l_r.h, l_r.p, \varepsilon) \bullet l_1, s_r, \varsigma)) \in p_1(x)$
- furthermore,

$$\lfloor (l_1, s, \varsigma) \bullet (l_2, s, \varsigma) \bullet (l_3, s, \varsigma) \rfloor = \lfloor ((l_r.h, l_r.p, \varepsilon) \bullet l_1, s_r, \varsigma) \bullet (l_2, s_r, \varsigma) \bullet (l_3, s_r, \varsigma) \rfloor$$

- hence, there exists $(l'_1, s', \varsigma'), (l'_2, s', \varsigma'), (l'_3, s', \varsigma') \in \mathcal{M}$, and $x' \in X$ such that

  $$(j-1, (l'_1, s', \varsigma')) \in q_1(x') \quad (j-1, (l'_2, s', \varsigma')) \in q_2(x') \quad (l_3, s_r, \varsigma) \; \hat{R}^{A\setminus\{t\}}_{j-1} \; (l'_3, s', \varsigma')$$

  and

  $$h' \in \lfloor (l'_1, s', \varsigma') \bullet (l'_2, s', \varsigma') \bullet (l'_3, s', \varsigma') \rfloor$$

- since $i \in \mathit{supp}_{A\setminus\{t\}}(q_1, q_2)$, we thus have that,

  $$(j-1, (\varepsilon, s'_r, \varsigma')) \in \boxed{q_1(x')}^{r,t,b} \qquad\qquad (j-1, (l'_2, s'_r, \varsigma')) \in q_2(x')$$

  where $s'_r = s[r \mapsto (l'_1, t, b)]$

- furthermore, as $s'(r) = s_r(r) = ((\varepsilon, \varepsilon, \pi_c(l_r)), t, b)$ it follows that

  $$h' \in \lfloor (\varepsilon, s'_r, \varsigma') \bullet (l'_2, s'_r, \varsigma') \bullet (l'_3, s'_r, \varsigma') \rfloor$$

- since

  $$i \in (\exists x : X. \; \boxed{p_1(x)}^{r,t,b} * p_2(x) \rightsquigarrow^{r,A\cup\{t\}}_{a} \exists x : X. \; \boxed{q_1(x)}^{r,t,b} * q_2(x))$$

  there exists an $\alpha \in \mathit{AId}$ such that,

  $$\pi_c(l_1 \bullet l_2)(r, \alpha) > 0 \qquad\qquad (j-1, s|_{A\cup\{n\}}, s'_r|_{A\cup\{t\}}) \in \varsigma(t)(b, r, \alpha)$$

- lastly, since $s_r \equiv_{A\setminus\{t\}} s$ and $s' \equiv_{A\setminus\{t\}} s'_r$ it thus follows from Lemma 63 that,

  $$(l_3, s, \varsigma) \; \hat{R}^{A\cup\{t\}}_{j-1} \; (l'_3, s'_r, \varsigma')$$

□

4.4.4 Atomic update allowed

Lemma 65.

$$\forall A \in \mathcal{P}(\mathit{RType}).\; \forall r \in \mathit{RId}.\; \forall a \in \mathit{Act}.\; \forall p_1, p_2, q_1, q_2 \in \mathit{Prop}.$$
$$(p_1 \rightsquigarrow^{r,A}_{a} q_1) \subseteq (p_1 * p_2 \rightsquigarrow^{r,A}_{a} q_1 * q_2)$$

Proof. Follows from the fact that $p * q \subseteq p$. □

Lemma 66.

$\forall I_p, I_q \in \mathit{Val} \times \mathit{AId} \times \mathit{Val} \to \mathit{Prop}.\; \forall t \in \mathit{RType}.\; \forall r \in \mathit{RId}.\; \forall a \in \mathit{Act}.\; \forall b, v \in \mathit{Val}.\; \forall \pi \in \mathsf{Perm}.$

$$\mathit{supp}_{A\cup\{t\}}(I_p(b,\alpha,v), I_q(b,\alpha,v)) \subseteq (\boxed{I_p(b,\alpha,v)}^{r,t,b}_{I_p,I_q} * [\alpha]^r_\pi \rightsquigarrow^{r,A\cup\{t\}}_{a} \boxed{I_q(b,\alpha,v)}^{r,t,b}_{I_p,I_q} * [\alpha]_\pi)$$

Proof.

- let $1 \leq j \leq i$, $(l_1, s_1, \varsigma_1), (l_2, s_2, \varsigma_2) \in \mathcal{M}$, $h, h' \in \mathit{Heap}$ such that

$$(j, (l_1, s_1, \varsigma_1)) \in \boxed{I_p(b, \alpha, v)}^{r,t,b}_{I_p, I_q} * [\alpha]^r_\pi$$
$$(j-1, (l_2, s_2, \varsigma_2)) \in \boxed{I_q(b, \alpha, v)}^{r,t,b}_{I_p, I_q} * [\alpha]^r_\pi$$

  and

$$h \in \lfloor (l_1, s_1, \varsigma_1) \rfloor \qquad h' \in \lfloor (l_2, s_2, \varsigma_2) \rfloor \qquad h' \in [\![a]\!](h)$$

- then $\varsigma_1(t)(b, r, \alpha)|_j = \mathit{act}(t, I_p, I_q)(b, r, \alpha)|_j$
- furthermore,

$$(j, (s_1(r).l, s_1, \varsigma_1)) \in I_p(b, \alpha, v) \qquad (j-1, (s_2(r).l, s_2, \varsigma_2)) \in I_q(b, \alpha, v)$$

- and, since $i \in \mathit{supp}_A(I_p(b, \alpha, v), I_q(b, \alpha, v))$,

$$(j, (s_1(r).l, s_1|_{A \cup \{t\}}, \varsigma_1)) \in I_p(b, \alpha, v) \quad (j-1, (s_2(r).l, s_2|_{A \cup \{t\}}, \varsigma_2)) \in I_q(b, \alpha, v)$$

- and thus, $(j-1, s_1|_{A \cup \{t\}}, s_2|_{A \cup \{t\}}) \in \mathit{act}(t, I_p, I_q)(b, r, \alpha)|_j = \varsigma_1(t)(b, r, \alpha)|_j$

$\square$

Points-to $\boxed{(-).(=) \mapsto (\equiv) : \Delta(\mathit{OId}) \times \Delta(\mathit{FName}) \times \Delta(\mathit{CVal}) \to \mathit{Prop} \in \text{ASets}}$

$$x.f \mapsto y \stackrel{\text{def}}{=} \{(i, m) \in \mathbb{N} \times \mathcal{M} \mid (m.l.h.o)(x, f) = y\}$$

Lemma 67.

$$\begin{aligned}
&\forall A, B \in \mathcal{P}(\mathit{RType}).\ \forall p_1, p_2, q_1, q_2, q_3 \in \mathit{Prop}.\ \forall t \in \mathit{RType}.\ \forall r \in \mathit{RId}.\\
&\forall a \in \mathit{Act}.\ \forall x \in \mathit{OId}.\ \forall b \in \mathit{Val}.\ \forall v_1, v_2 \in \mathit{CVal}.\ \forall f \in \mathit{FName}.\ v_1 \neq v_2 \ \Rightarrow\\
&\qquad \mathit{supp}_{A \setminus \{t\}}(p_1, p_2, q_1, q_2) \cap (a\ \mathit{sat}\ \{p_1 * p_2\}\{q_3\}) \cap\\
&\qquad (\mathit{valid}((q_1 * q_2) \Rightarrow x.f \mapsto v_1)) \cap (\mathit{valid}(q_3 \Rightarrow x.f \mapsto v_2))\\
&\qquad\quad \subseteq\ (\boxed{p_1}^{r,t,b} * p_2 \leadsto^{r,B}_a \boxed{q_1}^{r,t,b} * q_2)
\end{aligned}$$

Proof.

- let $j \in \mathbb{N}$, $m_1, m_2 \in \mathcal{M}$ and $h, h' \in \mathit{Heap}$ such that

$$(j, m_1) \in \boxed{p_1}^{r,t,b} * p_2 \qquad (j, m_2) \in \boxed{q_1}^{r,t,b} * q_2$$

  and

$$h \in \lfloor m_1 \rfloor \qquad h' \in \lfloor m_2 \rfloor \qquad h' \in [\![a]\!](h)$$

- since $j \in \mathit{supp}_{A \setminus \{t\}}(p_1, p_2, q_1, q_2)$ it thus follows that

$$(j, (m_1.l \bullet (m_1.s)(r).l, (m_1.s)[r \mapsto \bot], m_1.a)) \in p_1 * p_2$$
$$(j, (m_2.l \bullet (m_2.s)(r).l, (m_2.s)[r \mapsto \bot], m_2.a)) \in q_1 * q_2$$

  where $s[r \mapsto \bot]$ is notation for $s|_{dom(s) \setminus \{r\}}$

- and

$$\lfloor m_1 \rfloor = \lfloor (m_1.l \bullet (m_1.s)(r).l, (m_1.s)[r \mapsto \bot], m_1.a) \rfloor$$
$$\lfloor m_2 \rfloor = \lfloor (m_2.l \bullet (m_2.s)(r).l, (m_2.s)[r \mapsto \bot], m_2.a) \rfloor$$

- hence, there exists an $m_3 \in \mathcal{M}$ such that

$$(j-1, m_3) \in q_3 \qquad\qquad h' \in \lfloor m_3 \rfloor$$

- hence,

$$(j-1, (m_2.l \bullet (m_2.s)(r).l, (m_2.s)[r \mapsto \bot], m_2.a)) \in x.f \mapsto v_1$$
$$(j-1, m_3) \in x.f \mapsto v_2$$

- and thus $v_1 = h'(x, f) = v_2$, contradicting $v_1 \neq v_2$.

□

4.4.5 Non-expansiveness

Lemma 68.

$$\begin{aligned}
&\forall r \in \mathit{RId}.\ \forall t \in \mathit{RType}.\ \forall a \in \mathit{Val}.\ \forall \alpha \in \mathit{AId}.\\
&\forall i \in \mathbb{N}.\ \forall p_1, p_2, q_1, q_2 \in \mathit{Val} \times \mathit{AId} \times \mathit{Val} \to \mathit{Prop}.\\
&\qquad (\forall x \in \mathit{Val} \times \mathit{AId} \times \mathit{Val}.\ p_1(x) =^{\mathcal{V}}_{i+1} p_2(x) \wedge q_1(x) =^{\mathcal{V}}_{i+1} q_2(x)) \Rightarrow\\
&\qquad\qquad \mathit{act}(t, p_1, q_1)(a, r, \alpha)|_i \subseteq \mathit{act}(t, p_2, q_2)(a, r, \alpha)|_i
\end{aligned}$$

Proof.

- assume $(j, s_1, s_2) \in \mathit{act}(t, p_1, q_1)(a, r, \alpha)|_i$
- then $j \leq i$ and there exists $l_1, l_2 \in \mathit{LState}$, $\varsigma_1, \varsigma_2 \in \mathit{AMod}$, and $v \in \mathit{Val}$ such that

$$(j, (l_1, s_1, \varsigma_1)) \in p_1(a, \alpha, v) \quad (j, (l_2, s_2, \varsigma_2)) \in q_1(a, \alpha, v) \quad s_1(r) = (l_1, t, a) \quad s_2(r) = (l_2, t, a)$$

- hence,

$$(j, (l_1, s_1, a)) \in p_2(a, \alpha, v) \qquad\qquad (j, (l_2, s_2, a)) \in q_2(a, \alpha, v)$$

- and thus, $(j, s_1, s_2) \in act(t, p_2, q_2)(a, r, \alpha)|_i$

$\square$

Lemma 69.

$$\begin{aligned}&\forall t \in RId.\ \forall i \in \mathbb{N}.\ \forall p_1, p_2, q_1, q_2 \in Val \times AId \times Val \to Prop.\\ &\quad (\forall x \in Val \times AId \times Val.\ p_1(x) =_i^{\mathcal{V}} p_2(x) \wedge q_1(x) =_i^{\mathcal{V}} q_2(x)) \Rightarrow\\ &\qquad protocol(t, p_1, q_1) =_i^{\mathcal{V}} protocol(t, p_2, q_2)\end{aligned}$$

Proof.

- assume $j < i$ and $(j, m) \in protocol(t, p_1, q_1)$
- then
$$\forall x \in AArg.\ \pi_a(m)(t)(x)|_j = act(t, p_1, q_1)(x)|_j$$
- and by Lemma 68 and downwards-closure of $=_i^{\mathcal{V}}$,
$$\forall x \in Val \times AId \times Val.\ act(t, p_1, q_1)(x)|_j = act(t, p_2, q_2)(x)|_j$$
- hence $(j, m) \in protocol(t, p_2, q_2)$

$\square$

Lemma 70.

$$\forall i \in \mathbb{N}.\ \forall p_1, p_2, q_1, q_2 \in Prop.\ p_1 =_i^{\mathcal{V}} p_2 \wedge q_1 =_i^{\mathcal{V}} q_2 \wedge p_1 \sqsubseteq_i q_1 \Rightarrow p_2 \sqsubseteq_i q_2$$

Proof.

- let $r \in Prop$, $j \leq i$, $h \in Heap$, and $m, m_1, m_2 \in \mathcal{M}$ such that
$$m = m_1 \bullet m_2 \qquad (j, m_1) \in p_2 \qquad (j, m_2) \in r \qquad h \in \lfloor m \rfloor$$
- then $(j, m_1) \in p_1$ and there thus exists $m', m_1', m_2' \in \mathcal{M}$ such that
$$m' = m_1' \bullet m_2' \qquad (j, m_1') \in q_1 \qquad (j, m_2') \in r \qquad h \in \lfloor m' \rfloor$$
- hence, $(j, m_1') \in q_2$

$\square$

Lemma 71.

$$\begin{aligned}&\forall i \in \mathbb{N}.\ \forall a \in Act.\ \forall p_1, p_2, q_1, q_2 \in Prop.\\ &\quad p_1 =_i^{\mathcal{V}} p_2 \wedge q_1 =_i^{\mathcal{V}} q_2 \wedge a\ sat_i\ \{p_1\}\{q_1\} \Rightarrow a\ sat_i\ \{p_2\}\{q_2\}\end{aligned}$$

Lemma 72.

$$\begin{aligned}&\forall i \in \mathbb{N}.\ \forall x \in Thread.\ \forall p_1, p_2 \in Prop.\ \forall q_1, q_2 \in Stack \to Prop.\\ &\quad p_1 =_i^{\mathcal{V}} p_2 \wedge (\forall l \in Stack.\ q_1(l) =_i^{\mathcal{V}} q_2(l)) \wedge safe_i(x, p_1, q_1) \Rightarrow safe_i(x, p_2, q_2)\end{aligned}$$

5 Interpretation

In this section we define the interpretation of the logic in the model, and sketch the soundness of a few representative proof rules.

Types $\boxed{[\![\vdash \tau : \mathsf{Type}]\!] \in \text{ASets}}$

Types are interpreted as step-indexed equivalence relations. All base types except propositions and specifications have a plain identity as the step-indexed equivalence. Specifications and assertion propositions are considered i-equal if the agree up to level i. Products and function spaces are interpreted using the cartesian closed structure of ASets.

$$\begin{aligned}
[\![\vdash 1 : \mathsf{Type}]\!] &= 1\\
[\![\vdash \tau \to \sigma : \mathsf{Type}]\!] &= [\![\vdash \tau : \mathsf{Type}]\!] \to [\![\vdash \sigma : \mathsf{Type}]\!]\\
[\![\vdash \tau \times \sigma : \mathsf{Type}]\!] &= [\![\vdash \tau : \mathsf{Type}]\!] \times [\![\vdash \sigma : \mathsf{Type}]\!]\\
[\![\vdash \mathsf{Prop} : \mathsf{Type}]\!] &= \mathit{Prop}\\
[\![\vdash \mathsf{Spec} : \mathsf{Type}]\!] &= \mathit{Spec}\\
[\![\vdash \mathsf{Val} : \mathsf{Type}]\!] &= \Delta(\mathit{Val})\\
[\![\vdash \mathsf{Class} : \mathsf{Type}]\!] &= \Delta(\mathit{CName})\\
[\![\vdash \mathsf{Method} : \mathsf{Type}]\!] &= \Delta(\mathit{MName})\\
[\![\vdash \mathsf{Field} : \mathsf{Type}]\!] &= \Delta(\mathit{FName})\\
[\![\vdash \mathsf{Region} : \mathsf{Type}]\!] &= \Delta(\mathit{RId})\\
[\![\vdash \mathsf{Action} : \mathsf{Type}]\!] &= \Delta(\mathit{AId})\\
[\![\vdash \mathsf{RType} : \mathsf{Type}]\!] &= \Delta(\mathit{RType})\\
[\![\vdash \mathsf{Perm} : \mathsf{Type}]\!] &= \Delta((0,1])
\end{aligned}$$

Context $\boxed{[\![\Gamma]\!] \in \text{ASets}, [\![\Delta]\!] \in \text{ASets}}$

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

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

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

Terms are interpreted as morphisms in ASets and thus as non-expansive functions.

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

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

Specification logic

$$\begin{aligned}
[\![\Gamma \vdash \bot : \mathsf{Spec}]\!](\vartheta) &= \emptyset \\
[\![\Gamma \vdash \top : \mathsf{Spec}]\!](\vartheta) &= \mathbb{N} \\
[\![\Gamma \vdash \mathsf{S} \wedge \mathsf{T} : \mathsf{Spec}]\!](\vartheta) &= [\![\Gamma \vdash \mathsf{S} : \mathsf{Spec}]\!](\vartheta) \cap [\![\Gamma \vdash \mathsf{T} : \mathsf{Spec}]\!](\vartheta) \\
[\![\Gamma \vdash \mathsf{S} \vee \mathsf{T} : \mathsf{Spec}]\!](\vartheta) &= [\![\Gamma \vdash \mathsf{S} : \mathsf{Spec}]\!](\vartheta) \cup [\![\Gamma \vdash \mathsf{T} : \mathsf{Spec}]\!](\vartheta) \\
[\![\Gamma \vdash \mathsf{S} \Rightarrow \mathsf{T} : \mathsf{Spec}]\!](\vartheta) &= \{i \in \mathbb{N} \mid \forall j \le i. \\
&\qquad\quad j \in [\![\Gamma \vdash \mathsf{S} : \mathsf{Spec}]\!](\vartheta) \ \Rightarrow \\
&\qquad\quad j \in [\![\Gamma \vdash \mathsf{T} : \mathsf{Spec}]\!](\vartheta)\}
\end{aligned}$$

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

Separation logic

$$\begin{aligned}
[\![\Gamma;\Delta \vdash \mathsf{emp} : \mathsf{Prop}]\!](\vartheta,\delta) &= \mathit{emp} \\
[\![\Gamma;\Delta \vdash \mathsf{P} * \mathsf{Q} : \mathsf{Prop}]\!](\vartheta,\delta) &= [\![\Gamma;\Delta \vdash \mathsf{P} : \mathsf{Prop}]\!](\vartheta,\delta) * [\![\Gamma;\Delta \vdash \mathsf{Q} : \mathsf{Prop}]\!](\vartheta,\delta) \\
[\![\Gamma;\Delta \vdash \mathsf{P} \mathrel{-\!\!*} \mathsf{Q} : \mathsf{Prop}]\!](\vartheta,\delta) &= \{(i,m) \in \mathbb{N} \times \mathcal{M} \mid \forall j \leq i.\ \forall m' \geq m.\ \forall m'' \in \mathcal{M}. \\
&\qquad (m' \bullet m'' \text{ defined} \wedge (j,m'') \in [\![\Gamma;\Delta \vdash \mathsf{P} : \mathsf{Prop}]\!](\vartheta,\delta)) \\
&\qquad\quad \Rightarrow (j, m' \bullet m'') \in [\![\Gamma;\Delta \vdash \mathsf{Q} : \mathsf{Prop}]\!](\vartheta,\delta)\}
\end{aligned}$$

C♯

$$\begin{aligned}
[\![\Gamma;\Delta \vdash \mathsf{x} : \mathsf{Val}]\!](\vartheta,\delta) &= \delta(\mathsf{x}) \\
[\![\Gamma;\Delta \vdash \mathsf{null} : \mathsf{Val}]\!](\vartheta,\delta) &= \mathit{null} \\
[\![\Gamma;\Delta \vdash \mathsf{C} : \mathsf{Class}]\!](\vartheta,\delta) &= \mathsf{C} \\
[\![\Gamma;\Delta \vdash \mathsf{m} : \mathsf{Method}]\!](\vartheta,\delta) &= \mathsf{m} \\
[\![\Gamma;\Delta \vdash \mathsf{f} : \mathsf{Field}]\!](\vartheta,\delta) &= \mathsf{f}
\end{aligned}$$

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

Embeddings and guarded recursion

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

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

Region Types

$$\begin{aligned}
[\![\Gamma; \Delta \vdash \bot : \mathsf{RType}]\!](\vartheta, \delta) &= \text{the empty string} \\
[\![\Gamma; \Delta \vdash \mathsf{M} \leq \mathsf{N} : \mathsf{Prop}]\!](\vartheta, \delta) &= \{(i, m) \in \mathbb{N} \times \mathcal{M} \mid \\
&\qquad [\![\Gamma; \Delta \vdash \mathsf{N} : \mathsf{RType}]\!](\vartheta, \delta) \in ([\![\Gamma; \Delta \vdash \mathsf{M} : \mathsf{RType}]\!](\vartheta, \delta))^*\} \\
[\![\Gamma; \Delta \vdash \mathsf{M} \sqcap \mathsf{N} : \mathsf{RType}]\!](\vartheta, \delta) &= \text{longest common prefix of} \\
&\qquad [\![\Gamma; \Delta \vdash \mathsf{M} : \mathsf{RType}]\!](\vartheta, \delta) \text{ and } [\![\Gamma; \Delta \vdash \mathsf{N} : \mathsf{RType}]\!](\vartheta, \delta)
\end{aligned}$$

Concurrent abstract predicates

$$\begin{aligned}
[\![\Gamma; \Delta \vdash \boxed{\mathsf{P}}^{\mathsf{R},\mathsf{T},\mathsf{A}} : \mathsf{Prop}]\!](\vartheta, \delta) =\ & \mathit{region}([\![\Gamma; \Delta \vdash \mathsf{R} : \mathsf{Region}]\!](\vartheta, \delta), \\
& [\![\Gamma; \Delta \vdash \mathsf{T} : \mathsf{RType}]\!](\vartheta, \delta), \\
& [\![\Gamma; \Delta \vdash \mathsf{A} : \mathsf{Val}]\!](\vartheta, \delta), \\
& [\![\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop}]\!](\vartheta, \delta)) \\
[\![\Gamma; \Delta \vdash \mathsf{protocol}(\mathsf{T}, \mathsf{I}_p, \mathsf{I}_q) : \mathsf{Prop}]\!](\vartheta, \delta) =\ & \mathit{protocol}([\![\Gamma; \Delta \vdash \mathsf{T} : \mathsf{RType}]\!](\vartheta, \delta), \\
& [\![\Gamma; \Delta \vdash \mathsf{I}_p : \mathsf{Val} \times \mathsf{Action} \times \mathsf{Val} \to \mathsf{Prop}]\!](\vartheta, \delta), \\
& [\![\Gamma; \Delta \vdash \mathsf{I}_q : \mathsf{Val} \times \mathsf{Action} \times \mathsf{Val} \to \mathsf{Prop}]\!](\vartheta, \delta)) \\
[\![\Gamma; \Delta \vdash [\mathsf{A}]^{\mathsf{R}}_{\mathsf{P}} : \mathsf{Prop}]\!](\vartheta, \delta) =\ & \mathit{action}([\![\Gamma; \Delta \vdash \mathsf{A} : \mathsf{Action}]\!](\vartheta, \delta), \\
& [\![\Gamma; \Delta \vdash \mathsf{R} : \mathsf{Region}]\!](\vartheta, \delta), \\
& [\![\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Perm}]\!](\vartheta, \delta))
\end{aligned}$$

$$
\begin{aligned}
\llbracket \Gamma \vdash \mathsf{stable}(\mathsf{P}) : \mathsf{Spec} \rrbracket(\vartheta) &= \mathit{stable}(\llbracket \Gamma \vdash \mathsf{P} : \mathsf{Prop} \rrbracket(\vartheta)) \\
\llbracket \Gamma \vdash \mathsf{stable}^{\mathsf{R}}_{\mathsf{A}}(\mathsf{P}) : \mathsf{Spec} \rrbracket(\vartheta) &= \mathit{stable}^{\llbracket \Gamma \vdash \mathsf{R} : \mathsf{Region} \rrbracket(\vartheta)}_{\{\llbracket \Gamma \vdash \mathsf{A} : \mathsf{Action} \rrbracket(\vartheta)\}}(\llbracket \Gamma \vdash \mathsf{P} : \mathsf{Prop} \rrbracket(\vartheta)) \\
\llbracket \Gamma \vdash \mathsf{dep}_{\mathsf{T}}(\mathsf{P}) : \mathsf{Spec} \rrbracket(\vartheta) &= \mathit{supp}_{(\llbracket \Gamma \vdash \mathsf{T} : \mathsf{RType} \rrbracket(\vartheta))^*}(\llbracket \Gamma \vdash \mathsf{P} : \mathsf{Prop} \rrbracket(\vartheta)) \\
\llbracket \Gamma \vdash \mathsf{indep}_{\mathsf{T}}(\mathsf{P}) : \mathsf{Spec} \rrbracket(\vartheta) &= \mathit{supp}_{\mathit{RType} \setminus (\llbracket \Gamma \vdash \mathsf{T} : \mathsf{RType} \rrbracket(\vartheta))^*}(\llbracket \Gamma \vdash \mathsf{P} : \mathsf{Prop} \rrbracket(\vartheta)) \\
\llbracket \Gamma \vdash \mathsf{pure}_{\mathsf{protocol}}(\mathsf{P}) : \mathsf{Spec} \rrbracket(\vartheta) &= \mathit{pure}_{\mathit{protocol}}(\llbracket \Gamma \vdash \mathsf{P} : \mathsf{Prop} \rrbracket(\vartheta)) \\
\llbracket \Gamma \vdash \mathsf{pure}_{\mathsf{state}}(\mathsf{P}) : \mathsf{Spec} \rrbracket(\vartheta) &= \mathit{pure}_{\mathit{state}}(\llbracket \Gamma \vdash \mathsf{P} : \mathsf{Prop} \rrbracket(\vartheta)) \\
\llbracket \Gamma \vdash \mathsf{pure}_{\mathsf{perm}}(\mathsf{P}) : \mathsf{Spec} \rrbracket(\vartheta) &= \mathit{pure}_{\mathit{perm}}(\llbracket \Gamma \vdash \mathsf{P} : \mathsf{Prop} \rrbracket(\vartheta)) \\
\llbracket \Gamma \vdash \mathsf{P} \sqsubseteq \mathsf{Q} : \mathsf{Spec} \rrbracket(\vartheta) &= \llbracket \Gamma \vdash \mathsf{P} : \mathsf{Prop} \rrbracket(\vartheta) \sqsubseteq \llbracket \Gamma \vdash \mathsf{Q} : \mathsf{Prop} \rrbracket(\vartheta) \\
\llbracket \Gamma \vdash \mathsf{P} \sqsubseteq^{\mathsf{T}} \mathsf{Q} : \mathsf{Spec} \rrbracket(\vartheta) &= \llbracket \Gamma \vdash \mathsf{P} : \mathsf{Prop} \rrbracket(\vartheta) \sqsubseteq^{\mathit{RType} \setminus (\llbracket \Gamma \vdash \mathsf{T} : \mathsf{RType} \rrbracket(\vartheta))^*} \llbracket \Gamma \vdash \mathsf{Q} : \mathsf{Prop} \rrbracket(\vartheta) \\
\llbracket \Gamma \vdash \mathsf{P} \rightsquigarrow^{\mathsf{R},\mathsf{T}} \mathsf{Q} : \mathsf{Spec} \rrbracket(\vartheta) &= \{ i \in \mathbb{N} \mid \exists r \in \mathit{RId}.\ \exists t \in \mathit{RType}. \\
&\qquad\quad r = \llbracket \Gamma \vdash \mathsf{R} : \mathsf{Region} \rrbracket(\vartheta) \wedge \\
&\qquad\quad t = \llbracket \Gamma \vdash \mathsf{T} : \mathsf{RType} \rrbracket(\vartheta) \wedge \\
&\qquad\quad i \in (\llbracket \Gamma \vdash \mathsf{P} : \mathsf{Prop} \rrbracket(\vartheta) \rightsquigarrow^{r, \mathit{RType} \setminus t^*} \llbracket \Gamma \vdash \mathsf{Q} : \mathsf{Prop} \rrbracket(\vartheta)) \} \\
\llbracket \Gamma \vdash \mathsf{P} \rightsquigarrow^{\mathsf{R},\mathsf{T}}_{(\Delta).\langle \mathsf{s} \rangle} \mathsf{Q} : \mathsf{Spec} \rrbracket(\vartheta) &= \{ i \in \mathbb{N} \mid \exists p, q \in \llbracket \Delta \rrbracket \to \mathit{Prop}.\ \exists r \in \mathit{RId}.\ \exists t \in \mathit{RType}. \\
&\qquad \forall l, l' \in \mathit{Stack}.\ \forall a \in \mathit{Act}. \\
&\qquad\quad r = \llbracket \Gamma \vdash \mathsf{R} : \mathsf{Region} \rrbracket(\vartheta) \wedge \\
&\qquad\quad t = \llbracket \Gamma \vdash \mathsf{T} : \mathsf{RType} \rrbracket(\vartheta) \wedge \\
&\qquad\quad p = \lambda \delta \in \llbracket \Delta \rrbracket.\ \llbracket \Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop} \rrbracket(\vartheta, \delta) \wedge \\
&\qquad\quad q = \lambda \delta \in \llbracket \Delta \rrbracket.\ \llbracket \Gamma; \Delta \vdash \mathsf{Q} : \mathsf{Prop} \rrbracket(\vartheta, \delta) \wedge \\
&\qquad\quad (l, \mathsf{s}) \xrightarrow{a} (l', \varepsilon) \wedge \\
&\qquad\quad i \in (||p||_{\Delta}(l) \rightsquigarrow^{r, \mathit{RType} \setminus t^*}_{a} ||q||_{\Delta}(l')) \}
\end{aligned}
$$

Hoare assertions

$$
\begin{aligned}
&[\![\Gamma \vdash (\Delta).\{\mathsf{P}\}\bar{\mathsf{s}}\{\mathsf{Q}\}]\!](\vartheta) = \\
&\quad \{i \in \mathbb{N} \mid \exists p, q \in [\![\Delta]\!] \to \mathit{Prop}.\ \forall j \leq i.\ \forall t \in \mathit{TId}.\ \forall l \in \mathit{Stack}. \\
&\qquad\qquad p = \lambda\delta \in [\![\Delta]\!].\ [\![\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop}]\!](\vartheta, \delta) \wedge \\
&\qquad\qquad q = \lambda\delta \in [\![\Delta]\!].\ [\![\Gamma; \Delta \vdash \mathsf{Q} : \mathsf{Prop}]\!](\vartheta, \delta) \wedge \\
&\qquad\qquad \mathit{safe}_j((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 \{i \in \mathbb{N} \mid \exists p \in [\![\Delta, \mathtt{this}]\!] \to \mathit{Prop}.\ \exists q \in [\![\Delta, \mathtt{this}, \mathsf{r}]\!]. \\
&\qquad\quad \exists c \in \mathit{CName}.\ \exists m \in \mathit{MName}. \\
&\qquad\quad \forall j \leq i.\ \forall t \in \mathit{TId}.\ \forall l \in \mathit{Stack}.\ \forall o_t \in \mathit{OId}.\ \forall v_{\bar{\mathsf{y}}} \in \mathit{CVal}. \\
&\qquad\qquad p = \lambda\delta \in [\![\Delta, \mathtt{this}]\!].\ [\![\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop}]\!](\vartheta, \delta) \wedge \\
&\qquad\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\qquad c = [\![\Gamma; \Delta \vdash \mathsf{C} : \mathsf{Class}]\!](\vartheta) \wedge \\
&\qquad\qquad m = [\![\Gamma; \Delta \vdash \mathsf{M} : \mathsf{Method}]\!](\vartheta) \wedge \\
&\qquad\qquad \mathsf{body}(c, m) = \{\overline{\mathsf{Cy}}; \bar{\mathsf{s}}; \mathtt{return\ z}\} \wedge \\
&\qquad\qquad \mathit{safe}_j((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).\{\mathsf{P}\}\langle s \rangle\{\mathsf{Q}\} : \mathsf{Spec}]\!](\vartheta) = \\
&\quad \{i \in \mathbb{N} \mid \exists p, q \in [\![\Delta]\!] \to \mathit{Prop}.\ \forall j \leq i.\ \forall t \in \mathit{TId}.\ \forall l, l' \in \mathit{Stack}.\ \forall a \in \mathit{Act}. \\
&\qquad\qquad p = \lambda\delta \in [\![\Delta]\!].\ [\![\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop}]\!](\vartheta, \delta) \wedge \\
&\qquad\qquad q = \lambda\delta \in [\![\Delta]\!].\ [\![\Gamma; \Delta \vdash \mathsf{Q} : \mathsf{Prop}]\!](\vartheta, \delta) \wedge \\
&\qquad\qquad (l, s) \xrightarrow{a} (l', \varepsilon) \wedge \\
&\qquad\qquad j \in (a \text{ sat } \{||p||_\Delta(l)\}\{||q||_\Delta(l')\})\} \\
&[\![\Gamma \vdash (\Delta).\{\mathsf{P}\}\langle s \rangle^{\mathsf{T}}\{\mathsf{Q}\} : \mathsf{Spec}]\!](\vartheta) = \\
&\quad \{i \in \mathbb{N} \mid \exists p, q \in [\![\Delta]\!] \to \mathit{Prop}.\ \exists rt \in \mathit{RType}.\ \forall j \leq i.\ \forall t \in \mathit{TId}.\ \forall l, l' \in \mathit{Stack}.\ \forall a \in \mathit{Act}. \\
&\qquad\qquad p = \lambda\delta \in [\![\Delta]\!].\ [\![\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop}]\!](\vartheta, \delta) \wedge \\
&\qquad\qquad q = \lambda\delta \in [\![\Delta]\!].\ [\![\Gamma; \Delta \vdash \mathsf{Q} : \mathsf{Prop}]\!](\vartheta, \delta) \wedge \\
&\qquad\qquad rt = [\![\Gamma \vdash \mathsf{T} : \mathit{RType}]\!](\vartheta) \wedge \\
&\qquad\qquad (l, s) \xrightarrow{a} (l', \varepsilon) \wedge \\
&\qquad\qquad j \in (a \text{ sat}^{\mathit{RType} \setminus rt^*}\ \{||p||_\Delta(l)\}\{||q||_\Delta(l')\})\}
\end{aligned}
$$

where $|| - ||_\Delta : ([\![\Delta]\!] \to \mathcal{V}) \to (\mathit{Stack} \to \mathcal{V})$ 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 $\boxed{[\![\Gamma;\Delta \mid \Phi \mid \mathsf{P} \vdash \mathsf{Q}]\!] : 2}$

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

Specification logic entailment $\boxed{[\![\Gamma \mid \mathsf{S}_1, ..., \mathsf{S}_n \vdash \mathsf{T}]\!] : 2}$

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

5.1 Soundness

We have already proven the soundness of the most important proof rules for CAP, guarded recursion, the embeddings and phantom state in Section 4. In this section we consider the soundness of the Hoare logic. In particular, we prove the soundness of two representative proof rules, namely the frame rule (Lemma 76) and the proof rule for forking a new thread (Lemma 78).

Since framing of assertions is explicitly build into the interpretation of the *safe* predicate, the soundness of the frame rule reduces to proving that if $\mathsf{mod}(\bar{\mathsf{s}}) \cap \mathsf{FV}(\mathsf{R}) = \emptyset$ then the interpretation of R is the same in initial and terminal stack of $\bar{\mathsf{s}}$. Lemma 75 expresses that mod is an over-approximation of the set of modified stack variables. Note that mod is only defined on sequences of statements (i.e., mod is *not* defined on thread call stacks) and this lemma is explicitly stated in terms of a sequence of statements. Since method and delegate calls introduce a new stack frame, which is restored upon their return, we can ignore the bodies of method and delegate calls. This is achieved using Lemma 74, which allows us strengthen the post-condition of $s_1; s_2$, by strengthening the post-condition of s_2 (under an arbitrary pre-condition).

Lemma 73 (FV). *If* $\Gamma;\Delta \vdash \mathsf{M} : \tau$ *then*

$$\begin{aligned}&\forall\vartheta \in [\![\Gamma]\!].\ \forall\delta_1,\delta_2 \in [\![\Delta]\!]. \\ &\quad (\forall x \in \mathsf{FV}(\mathsf{M}).\ \delta_1(x) = \delta_2(x)) \Rightarrow [\![\Gamma;\Delta \vdash \mathsf{M} : \tau]\!](\vartheta,\delta_1) = [\![\Gamma;\Delta \vdash \mathsf{M} : \tau]\!](\vartheta,\delta_2)\end{aligned}$$

Lemma 74.

$$\begin{aligned}&\forall i \in \mathbb{N}.\ \forall t \in \mathit{TId}.\ \forall l \in \mathit{Stack}.\ \forall s_1, s_2 \in \mathit{TCStack}.\ \forall p \in \mathit{Prop}.\ \forall q, q' \in \mathit{Stack} \to \mathit{Prop}. \\ &\quad (\forall j \leq i.\ \forall r \in \mathit{Prop}.\ \forall l' \in \mathit{LState}.\ \mathit{safe}_j((t, l', s_2), r, q) \Rightarrow \mathit{safe}_j((t, l', s_2), r, q')) \Rightarrow \\ &\qquad \mathit{safe}_i((t, l, s_1; s_2), p, q) \Rightarrow \mathit{safe}_i((t, l, s_1; s_2), p, q')\end{aligned}$$

Proof. By induction on i. As safety is trivial for $i = 0$, assume $i = j + 1$.

- Case $irr(s_1; s_2)$ or $s_1 = \varepsilon$:
  - then $s_1 = \varepsilon$
  - hence $\mathit{safe}_i((t, l, s_2), p, q)$ and thus
$$\mathit{safe}_i((t, l, s_2), p, q')$$
- Case $(t, l, s_1; s_2) \xrightarrow{a} \{(t, l', s')\} \uplus T$ and $s_1 \neq \varepsilon$:
  - by Lemma 15 there thus exists an s'_1 such that $s' = s'_1; s_2$
  - furthermore, by safety there exists a
$$p' \in \{(t, l', s'_1; s_2)\} \uplus T \to \mathit{Prop}$$
    such that
$$\begin{array}{l} \forall z \in \{(t, l', s'_1; s_2)\} \uplus T.\ i \in \mathit{stable}(p'(z)) \\ a\ \mathit{sat}_i\ \{p\}\ \{p'(t, l', s'_1; s_2) * \circledast_{z \in T} p'(z)\} \\ \mathit{safe}_j((t, l', s'_1; s_2), p'(t, l', s'_1; s_2), q) \\ \forall z \in T.\ \mathit{safe}_j(z, p'(z), \lambda\_.\ \top) \end{array}$$
  - hence, by the induction hypothesis,
$$\mathit{safe}_j((t, l', s'_1; s_2), p'(t, l', s'_1; s_2), q')$$

□

Lemma 75 (Modifies clause).

$\forall i \in \mathbb{N}.\ \forall t \in \mathit{TId}.\ \forall l \in \mathit{Stack}.\ \forall s \in \mathit{seq\ Stm}.\ \forall p \in \mathit{Prop}.\ \forall q \in \mathit{Stack} \to \mathit{Prop}.\ \forall x \in \mathit{Var}.$
$x \in \mathit{dom}(l) \wedge x \notin \mathsf{mod}(s) \wedge \mathit{safe}_i((t, l, \mathit{stm}(s)), p, q) \Rightarrow$
$\mathit{safe}_i((t, l, \mathit{stm}(s)), p, \lambda l' \in \mathit{LState}.\ q(l') \wedge l(x) = l'(x))$

Proof. By induction on i. As safety is trivial for $i = 0$, assume $i = j + 1$.

- Case $irr(t, l, s)$:
  - by assumption
$$p \sqsubseteq_i q(l)$$
  - and clearly,
$$q(l) = (q(l) \wedge l(x) = l(x))$$
- Case $(t, l, s) \xrightarrow{a} \{(t, l', s')\} \uplus T$:
  - Case Seq followed by MCall:

* there exists $y, z, \bar{u}, r \in \mathit{Var}$ and $s_b, s'' \in \mathit{seq\ Stm}$ such that

$$s = (y = z.\mathsf{m}(\bar{u}); s'') \qquad\qquad s' = s_b; \mathit{return}\ (l, y, r); s''$$

and $T = \emptyset$
* since $x \notin \mathsf{mod}(s)$, $x \neq y$ and $x \notin \mathsf{mod}(s'')$
* by safety of s there exists a $p' \in \mathit{Prop}$ such that

$$i \in \mathit{stable}(p') \qquad\qquad i \in (a\ \mathit{sat}\ \{i\}\{p\}p')$$

$$\mathit{safe}_j((t, l', s_b; \mathit{return}\ (l, y, r); s''), p', q)$$

* by Lemma 74 it thus suffices to prove

$\forall k \leq j.\ \forall p'' \in \mathit{Prop}.\ \forall l' \in \mathit{LState}.$
$\quad \mathit{safe}_k((t, l', \mathit{return}\ (l, y, r); s''), p'', q) \ \Rightarrow$
$\quad\quad \mathit{safe}_k((t, l', \mathit{return}\ (l, y, r); s''), p'', \lambda l' \in \mathit{LState}.\ q(l') \wedge l(x) = l'(x))$

Case $(t, l', \mathit{return}\ (l, y, r); s'') \xrightarrow{id} (t, l[y \mapsto l'(r)], s'')$:

· by assumption there thus exists a $p''' \in \mathit{Prop}$ such that

$$k \in \mathit{stable}(p''') \qquad\qquad k \in (id\ \mathit{sat}\ \{p''\}\{p'''\})$$

$$\mathit{safe}_{k-1}((t, l[y \mapsto l'(r)], s''), p''', q)$$

· hence, by the induction hypothesis (as $x \notin \mathsf{mod}(s'')$):

$$\mathit{safe}_{k-1}((t, l[y \mapsto l'(r)], s''), p''', \lambda l' \in \mathit{LState}.\ q(l') \wedge (l[y \mapsto l'(r)])(x) = l'(x))$$

· hence, as $x \neq y$:

$$\mathit{safe}_{k-1}((t, l[y \mapsto l'(r)], s''), p''', \lambda l' \in \mathit{LState}.\ q(l') \wedge l(x) = l'(x))$$

– Case Seq followed by DCall: same as above
– Case Seq followed by Assign/FRead/FWrite/IfT/IfF/CAlloc/DAlloc:
* then there exists $s_1 \in \mathit{Stm}$ and $s_1', s_2 \in \mathit{seq\ Stm}$ such that

$$s = s_1; s_2 \qquad s' = s_1'; s_2 \qquad x \notin \mathsf{mod}(s') \qquad T = \emptyset$$

* by safety there thus exists a $p' \in \mathit{Prop}$ such that

$$i \in \mathit{stable}(p) \qquad\qquad i \in (a\ \mathit{sat}\ \{p\}\{p'\})$$

$$\mathit{safe}_j((t, l', s'), p', q)$$

    - $*$ and, by the induction hypothesis,

$$\mathit{safe}_j((t, l', s'), p', \lambda l'' \in \mathit{LState}.\ q(l') \wedge l'(x) = l''(x))$$

    - $*$ since $x \notin \mathsf{mod}(s)$, $l(x) = l'(x)$ and thus,

$$\mathit{safe}_j((t, l', s'), p', \lambda l'' \in \mathit{LState}.\ q(l') \wedge l(x) = l''(x))$$

  – Case Fork: similar to the above cases

$\square$

Lemma 76 (Frame rule)**.** *The following rule is sound.*

$$\frac{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}\}\bar{\mathsf{s}}\{\mathsf{Q}\} \qquad \mathsf{mod}(\bar{\mathsf{s}}) \cap \mathsf{FV}(\mathsf{R}) = \emptyset}{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P} * \mathsf{R}\}\bar{\mathsf{s}}\{\mathsf{Q} * \mathsf{R}\}}$$

Proof.

- suppose $\vartheta \in [\![\Gamma]\!]$ and $i \in [\![\Phi]\!]$
- as safety is trivial for $i = 0$, assume $i = j + 1$
- let

$$\begin{aligned} p &= \lambda\delta \in [\![\Delta]\!].\ [\![\Gamma; \Delta \vdash \mathsf{P} : \mathsf{Prop}]\!](\vartheta, \delta) \\ q &= \lambda\delta \in [\![\Delta]\!].\ [\![\Gamma; \Delta \vdash \mathsf{Q} : \mathsf{Prop}]\!](\vartheta, \delta) \\ r &= \lambda\delta \in [\![\Delta]\!].\ [\![\Gamma; \Delta \vdash \mathsf{R} : \mathsf{Prop}]\!](\vartheta, \delta) \end{aligned}$$

- suppose $t \in \mathit{TId}$ and $l \in \mathit{Stack}$
- by assumption,

$$\mathit{safe}_i((t, l, \bar{\mathsf{s}}), ||p||_\Delta(l), ||q||_\Delta)$$

- hence, by Lemma 14,

$$\mathit{safe}_i((t, l, \bar{\mathsf{s}}), ||p||_\Delta(l) * ||r||_\Delta(l), \lambda l' \in \mathit{Stack}.\ ||q||_\Delta(l') * ||r||_\Delta(l))$$

- since $\mathsf{mod}(\mathsf{s}) \cap \mathsf{FV}(\mathsf{R}) = \emptyset$ it thus follows by Lemma 75 that,

$$\mathit{safe}_i((t, l, \bar{\mathsf{s}}), ||p||_\Delta(l) * ||r||_\Delta(l), \lambda l' \in \mathit{Stack}.\ ||q||_\Delta(l') * ||r||_\Delta(l) \wedge \bigwedge_{x \in \mathsf{FV}(\mathsf{R})} l'(x) = l(x))$$

- hence, by Lemma 73,

$$\mathit{safe}_i((t, l, \bar{\mathsf{s}}), ||p||_\Delta(l) * ||r||_\Delta(l), \lambda l' \in \mathit{Stack}.\ ||q||_\Delta(l') * ||r||\Delta(l'))$$

$\square$

Lemma 77.

$$\forall i \in \mathbb{N}.\ \forall t \in \mathit{TId}.\ \forall l \in \mathit{Stack}.\ \forall s \in \mathit{seq\ Stm}.\ \forall p \in \mathit{Prop}.\ \forall q \in \mathit{Stack} \to \mathit{Prop}.$$
$$\mathit{safe}_i((t, l, \mathit{stm}(s)), p, q) \Rightarrow \mathit{safe}_i((t, l, \mathit{stm}(s)), p, \lambda\_ .\ \top)$$

Lemma 78 (Forking)**.** *The following proof rule is sound.*

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

Proof.

- suppose $\vartheta \in [\![\Gamma]\!]$, $i \in [\![\Phi]\!]$, $t \in \mathit{TId}$, and $l \in \mathit{Stack}$
- as safety is trivial if $i = 0$ assume $i = j + 1$
- suppose $(t, l, \mathtt{fork}(\mathsf{x})) \xrightarrow{a} \{(t, l', s')\} \uplus T$
- then there exists $\mathsf{C} \in \mathit{CName}$, $o \in \mathit{OId}$, $\mathsf{m}_b \in \mathit{MName}$, $\mathsf{r}, \bar{\mathsf{y}} \in \mathit{Var}$, $l'' \in \mathit{Stack}$, $t' \in \mathit{TId}$, $\bar{\mathsf{s}} \in \mathit{Stm}$ such that,

$$a = \mathit{ctype}(l(\mathsf{x}), \mathsf{C}, o, \mathsf{m}_b) \qquad T = \{(t', l'', \bar{\mathsf{s}})\} \qquad l'' = [\mathtt{this} \mapsto o, \bar{\mathsf{y}} \mapsto \mathit{null}]$$

  and

$$\mathsf{body}(\mathsf{C}, \mathsf{m}_b) = \mathtt{void}\ \mathsf{m}_b()\{\overline{\mathsf{Cy}}; \bar{\mathsf{s}}; \mathtt{return}\ \mathsf{r}\} \qquad l' = l \qquad s' = \varepsilon$$

- let

$$\begin{aligned}
C &= [\![\Gamma \vdash \mathsf{C} : \mathsf{Class}]\!](\vartheta)\\
m &= [\![\Gamma \vdash \mathsf{m} : \mathsf{Method}]\!](\vartheta)\\
p_1 &= \lambda\delta \in [\![\mathtt{this}]\!].\ [\![\Gamma; \mathtt{this} \vdash \mathsf{P} : \mathsf{Prop}]\!](\vartheta, \delta)\\
q_1 &= \lambda\delta \in [\![\mathtt{this}, \mathsf{r}]\!].\ [\![\Gamma; \mathtt{this}, \mathsf{r} \vdash \mathsf{Q}[\mathsf{r}/\mathsf{ret}] : \mathsf{Prop}]\!](\vartheta, \delta)\\
p_2 &= \lambda\delta \in [\![\Delta]\!].\ [\![\Gamma; \Delta \vdash \mathsf{P}[\mathsf{y}/\mathtt{this}] : \mathsf{Prop}]\!](\vartheta, \delta)\\
p_3 &= \lambda\delta \in [\![\Delta]\!].\ [\![\Gamma; \Delta \vdash \mathsf{x} \mapsto \mathsf{y.m} * \mathsf{y} : \mathsf{C} : \mathsf{Prop}]\!](\vartheta, \delta)\\
p_4 &= \lambda\delta \in [\![\Delta]\!].\ [\![\Gamma; \Delta \vdash \mathsf{P}[\mathsf{y}/\mathtt{this}] * \mathsf{x} \mapsto \mathsf{y.m} * \mathsf{y} : \mathsf{C} : \mathsf{Prop}]\!](\vartheta, \delta)\\
&= \lambda\delta \in [\![\Delta]\!].\ p_2(\delta) * p_3(\delta)\\
q_2 &= \lambda\delta \in [\![\Delta]\!].\ [\![\Gamma; \Delta \vdash \mathsf{emp} : \mathsf{Prop}]\!](\vartheta, \delta)\\
&= \lambda\delta \in [\![\Delta]\!].\ \mathit{emp}
\end{aligned}$$

- then

$$i \in (\mathit{ctype}(l(\mathsf{x}), \mathsf{C}, o, \mathsf{m}_b)\ \mathit{sat}\ \{||p_2||_\Delta(l)\}\{||p_2||_\Delta(l) * l(\mathsf{y}) = o * \mathsf{m}_b = m * \mathsf{C} = C\})$$

- suppose $l(\mathsf{y}) = o$, $\mathsf{m}_b = m$, and $\mathsf{C} = C$ (safety follows trivially otherwise)
- hence, as $l(\mathsf{y}) = o$,
$$||p_2||_\Delta(l) = ||p_1||_{\texttt{this}}(l'')$$
- furthermore, by assumption,
$$i \in \mathit{stable}(||p_2||_\Delta(l)) \qquad \mathit{safe}_j((t', l'', s_b), ||p_1||_{\texttt{this}}(l''), ||q||_{\texttt{this},\mathsf{r}})$$
- and thus
$$\mathit{safe}_j((t', l'', s_b), ||p_2||_\Delta(l), ||q||_{\texttt{this},\mathsf{r}})$$
- hence, by Lemma 77,
$$\mathit{safe}_j((t', l'', s_b), ||p_2||_\Delta(l), \lambda\_.\ \top)$$
- and
$$\mathit{safe}_j((t, l, \varepsilon), l(\mathsf{y}) = o * m_b = m * C = T, ||q_2||_\Delta)$$
as $p \sqsubseteq \mathit{emp}$ for all $p \in \mathit{Prop}$

$\square$

Theorem 2 (Soundness)**.** *The specification and assertion logic is sound:*

If $\Gamma; \Delta \mid \Phi \mid \mathsf{P} \vdash \mathsf{Q}$ *then* $[\![\Gamma; \Delta \mid \Phi \mid \mathsf{P} \vdash \mathsf{Q}]\!]$.
If $\Gamma \vdash \Phi \vdash \mathsf{S}$ *then* $[\![\Gamma \vdash \Phi \vdash \mathsf{S}]\!]$.

Theorem 3. *If* $\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}\}\bar{\mathsf{s}}\{\mathsf{Q}\}$ *then for all*

$$\vartheta \in [\![\Gamma]\!] \qquad i \in [\![\Phi]\!](\vartheta) \qquad l \in [\![\Delta]\!]$$

and

$$t \in \mathit{TId} \qquad j \le i \qquad h \in \lfloor ||\lambda\delta.\ [\![\Gamma; \Delta \vdash \mathsf{P} : \mathit{Prop}]\!](\vartheta, \delta)||_\Delta(l) \rfloor_i$$

if

$$(h, \{(t, l, \mathit{stm}(\bar{\mathsf{s}}))\}) \to^j (h', \{(t, l', \mathit{skip})\} \uplus T')$$

and T' *is irreducible then* $h' \in \lfloor ||\lambda\delta.\ [\![\Gamma; \Delta \vdash \mathsf{Q} : \mathit{Prop}]\!](\vartheta, \delta)||_\Delta(l') \rfloor_{i-j}$.

References

[1] L. Birkedal, R. Mogelberg, J. Schwinghammer, and K. Stovring. First steps in synthetic guarded domain theory: step-indexing in the topos of trees. In *Proceedings of LICS*, 2011.

[2] L. Birkedal, B. Reus, J. Schwinghammer, K. Stovring, J. Thamsborg, and H. Yang. Step-indexed Kripke models over recursive worlds. In *Proceedings of POPL*, 2011.

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

[4] J. Schwinghammer, L. Birkedal, B. Reus, and H. Yang. Nested Hoare triples and frame rules for higher-order store. In *Proceedings of CSL*, Apr. 2009.

[5] J. Schwinghammer, L. Birkedal, B. Reus, and H. Yang. Nested Hoare Triples and Frame Rules for Higher-Order Store. *LMCS*, 7(3:21), 2011.

[6] K. Svendsen, L. Birkedal, and M. Parkinson. A Specification of the Joins Library in Higher-order Separation Logic, 2012.

[7] K. Svendsen, L. Birkedal, and M. Parkinson. Verification of the joins library in higher-order separation logic. Technical report, IT University of Copenhagen, 2012. Available at `www.itu.dk/people/kasv/joins-tr.pdf`.