A Separation Logic for Fictional Sequential Consistency: Technical Appendix

Filip Sieczkowski
IT University of Copenhagen
fisi@cs.au.dk

Kasper Svendsen
Aarhus University
ksvendsen@cs.au.dk

Lars Birkedal
Aarhus University
birkedal@cs.au.dk

Jean Pichon-Pharabod
University of Cambridge
Jean.Pichon@cl.cam.ac.uk

April 14, 2015

Contents

1 The Language 3

2 The Model 6
2.1 Local States, Shared States and the Treatment of Store Buffers 6
2.2 Interference Relation 9
2.3 Region Interpretations 9
2.4 Instrumented States 10
2.5 Logical Connectives 11
2.6 Stability 14
2.7 View Shift 14
2.8 Atomic Satisfaction 15
2.9 Specification Embedding 16
2.10 Thread Safety 17
2.11 Thread-pool Evaluation 24

3 Logic 26
3.1 TSO Assertion Logic 27
3.2 Syntactic Sugar 27
3.3 Read and Flush Rules 27
3.4 Specification Logic 28
3.4.1 Reasoning about Hoare triples 29
3.4.2 Later operator 32
3.4.3 Reasoning about stability 32
3.4.4 Method verification 33

4 Interpretation 34
4.1 Types . . . 34
4.2 Contexts . . . 34
4.3 Lambda calculus . . . 34
4.4 TSO Assertion Logic . . . 34
4.5 SC Assertion Logic . . . 35
4.6 Specification Logic . . . 36
4.7 Read and Flush Judgments . . . 37
4.8 Embeddings and later operators . . . 37
4.9 Entailment . . . 37

5 Soundness 37
5.1 Read and Flush Judgments . . . 43

6 Verification of a spin-lock in the TSO logic 45
6.1 Specification . . . 45
6.2 Predicate Definitions . . . 46
6.3 Proof outline . . . 46

7 Logical Atomicity: Treiber's Stack 50
7.1 Specification . . . 50
7.2 Predicate definitions . . . 51
7.3 Proof outline . . . 52
7.4 Deriving an SC shared bag specification . . . 55

8 Verification of a bounded ticket lock in the TSO logic 57
8.1 Specification . . . 57
8.2 Discussion of the design choices . . . 58
8.3 acquire version . . . 59
8.4 Definitions . . . 60
8.5 Proof outline . . . 61
8.5.1 Outline . . . 61
8.5.2 Discussion . . . 62
8.6 Proof . . . 63
8.6.1 ticket . . . 64
8.6.2 lock . . . 65
8.6.3 release . . . 66
8.6.4 Constructor . . . 67

9 Verification of a double-checked initialisation wrapper in the TSO logic 68
9.1 Specification . . . 68
9.2 Definitions . . . 69
9.3 Proof outline . . . 70
9.3.1 Constructor . . . 70

9.3.2 get . . . 70
9.3.3 Duplicability . . . 71
9.3.4 Equality of results . . . 71
9.4 Proof detail . . . 71

10 Verification of a circular buffer in an extension of the TSO logic 73
10.1 Treatment of arrays . . . 73
10.2 Specification . . . 73
10.3 Definitions . . . 74
10.4 Proof outline . . . 77

References 80

1 The Language

We build our logic for a simple, class-based programming language. For simplicity of both semantics and the logic, the language uses let-bindings and expressions, but it stays relatively low-level by ensuring that all the values are machine-word size. The values include variables, natural numbers, booleans, unit, object references (pointers), the null pointer and the special variable **this**. The expressions include values, let bindings, conditionals, constructor and method calls, field reads and writes, atomic compare-and-swap expression and a fork call. The syntax of values and expressions is shown in Figure 1. Note that for simplicity classes can have only one constructor each, and that it has to end with **this**, which ensures that constructors return the newly allocated value. Actual object references o cannot occur in the text of the program, but need to be values for the operational semantics to be able to reduce.

$$
\begin{array}{rll}
\mathsf{ClassDef} ::= & \mathbf{class}\ \mathsf{C} = \{\overline{\mathsf{T}\ \mathsf{f}}; \mathsf{CDef}; \overline{\mathsf{MDef}}\} & \text{(Class definition)} \\
\mathsf{MDef} ::= & \mathsf{T}\ \mathsf{m}(\overline{\mathsf{T}\ \mathsf{x}}) = \mathsf{e} & \text{(Method definition)} \\
\mathsf{CDef} ::= & \mathsf{C}(\overline{\mathsf{T}\ \mathsf{x}}) = (\mathsf{e}; \mathbf{this}) & \text{(Constructor definition)} \\
\mathsf{Val} \ni \mathsf{v} ::= & \mathsf{x} \mid \mathbf{null} \mid \mathbf{this} \mid \mathsf{o} \mid \mathsf{n} \mid \mathsf{b} \mid \mathbf{()} & \text{(Values)} \\
\mathsf{Expr} \ni \mathsf{e} ::= & \mathsf{v} \mid \mathbf{let}\ \mathsf{x} = \mathsf{e}_1\ \mathbf{in}\ \mathsf{e}_2 \mid \mathbf{if}\ \mathsf{v}\ \mathbf{then}\ \mathsf{e}_1\ \mathbf{else}\ \mathsf{e}_2 & \text{(Expressions)} \\
 & \mid \mathbf{new}\ \mathsf{C}(\overline{\mathsf{v}}) \mid \mathsf{v.f} \mid \mathsf{v.f} := \mathsf{v} \mid \mathsf{v.m}(\overline{\mathsf{v}}) \mid \mathbf{CAS}(\mathsf{v}_1.\mathsf{f}, \mathsf{v}_2, \mathsf{v}_3) \mid \mathbf{fork}(\mathsf{v.m}) \mid \mathbf{fence} & \\
\mathsf{E} ::= & [\,] \mid \mathbf{let}\ \mathsf{x} = \mathsf{E}\ \mathbf{in}\ \mathsf{e} & \text{(Evaluation Contexts)}
\end{array}
$$

Figure 1: The syntax of the programming language.

In Figure 2 we present the semantic domains: the interesting part is the machine state, which in addition to the heap also contains a pool of store buffers, one per thread, and the program state. The latter is only consistent if the thread pool contains the same

threads that the store buffer pool contains — or if the program has already gone wrong.

$$
\begin{array}{rll}
t \in \mathit{TId} & & \text{(Thread Identifiers)} \\
o \in \mathit{OId} & & \text{(Object Identifiers)} \\
v \in \mathit{Val} & \stackrel{\text{def}}{=} \{\textbf{null}\} + \mathit{OId} + \mathbb{N} + \mathbf{2} + \mathbf{1} & \text{(Semantic Values)} \\
\mathit{OHeap} & \stackrel{\text{def}}{=} \mathit{OId} \times \mathit{FName} \rightharpoonup_{fin} \mathit{Val} & \text{(Object Heaps)} \\
\mathit{THeap} & \stackrel{\text{def}}{=} \mathit{OId} \rightharpoonup_{fin} \mathit{CName} & \text{(Type Heaps)} \\
h \in \mathit{Heap} & \stackrel{\text{def}}{=} \mathit{OHeap} \times \mathit{THeap} & \text{(Heaps)} \\
s \in \mathit{SBuffer} & \stackrel{\text{def}}{=} \mathrm{seq}\,(\mathit{OId} \times \mathit{FName} \times \mathit{Val}) & \text{(Store Buffers)} \\
T \in \mathit{TPool} & \stackrel{\text{def}}{=} \mathit{TId} \rightharpoonup_{fin} \mathit{Expr} & \text{(Thread Pool)} \\
U \in \mathit{SPool} & \stackrel{\text{def}}{=} \mathit{TId} \rightharpoonup_{fin} \mathit{SBuffer} & \text{(Store Buffer Pool)} \\
\mu \in \mathit{MSt} & \stackrel{\text{def}}{=} \mathit{Heap} \times \mathit{SPool} + \{\lightning\} & \text{(Memory State)} \\
\mathit{PSt} & \stackrel{\text{def}}{=} \{(\mu, T) \in \mathit{MSt} \times \mathit{TPool} \mid & \text{(Program State)} \\
& \quad \forall h \in \mathit{Heap}, U \in \mathit{SPool}.\ \mu = (h, U) \Rightarrow \operatorname{dom} U = \operatorname{dom} T &
\end{array}
$$

Figure 2: Semantic domains

Following the Views framework [2], the operational semantics is split into two components: a thread-local small-step semantics labeled with actions that occur during the step, and action semantics that defines the effect of the action on the machine state. Below, the possible actions are given, and in the Figure 3 we present the thread-local semantics. The interpretation of actions is shown in Figure 6.

$$
\begin{array}{rcl}
a \in \mathit{Act} & ::= & \varepsilon \mid \mathit{read}(t, o, f, v) \mid \mathit{write}(t, o, f, v) \mid \mathit{cas}(t, o, f, v_o, v_n, r) \mid \mathit{new}(t, C, o) \\
& \mid & \mathit{type}(t, o, T) \mid \mathit{fork}(t, o, T, t') \mid \mathit{fence}(t) \mid \lightning
\end{array}
$$

Note that the action semantics uses two helper functions: lookup and flush. These are used to determine the result of reading a field from a thread's perspective, and to update the state of the heap with a contents of the buffer. These functions are defined in Figure 5.

Finally, the complete semantics proceeds by reducing one of the threads using the thread-local semantics, then interpreting the resulting action with the action semantics, and reducing to a memory state in the resultant set, as in Figure 4. Note how in some cases, notably read, this might require "guessing" the return value, and checking that the guess was right using the action semantics.

Finally, we show one of the simple lemmas about operational semantics, that we need in the process of building the logic. The decomposition lemma states that the only

$$\frac{}{(t, E[\textbf{let}\ \mathsf{x} = \mathsf{v}\ \textbf{in}\ \mathsf{e}]) \xrightarrow{\varepsilon} \{(t, E[\mathsf{e}[\mathsf{v}/\mathsf{x}]])\}}\ \textsc{Let}$$

$$\frac{}{(t, E[\textbf{if true then}\ \mathsf{e}_1\ \textbf{else}\ \mathsf{e}_2]) \xrightarrow{\varepsilon} \{(t, E[\mathsf{e}_1])\}}\ \textsc{If-True}$$

$$\frac{}{(t, E[\textbf{if false then}\ \mathsf{e}_1\ \textbf{else}\ \mathsf{e}_2]) \xrightarrow{\varepsilon} \{(t, E[\mathsf{e}_2])\}}\ \textsc{If-False}$$

$$\frac{}{(t, E[\textbf{CAS}(\mathsf{o.f}, \mathsf{v_o}, \mathsf{v_n})]) \xrightarrow{cas(t,\mathsf{o},\mathsf{f},\mathsf{v_o},\mathsf{v_n},\mathsf{r})} \{(t, E[\mathsf{r}])\}}\ \text{CAS}$$

$$\frac{\mathsf{ctorBody}(\mathsf{C}) = (\mathsf{C}(\overline{\mathsf{T}\ \mathsf{x}}) = \mathsf{e})}{(t, E[\textbf{new}\ \mathsf{C}(\bar{\mathsf{v}})]) \xrightarrow{new(t,\mathsf{C},\mathsf{o})} \{(t, E[\mathsf{e}[\mathsf{o}, \bar{\mathsf{v}}/\textbf{this}, \bar{\mathsf{x}}]])\}}\ \textsc{New}$$

$$\frac{}{(t, E[\mathsf{o.f}]) \xrightarrow{read(t,\mathsf{o},\mathsf{f},\mathsf{v})} \{(t, E[\mathsf{v}])\}}\ \textsc{Read}$$

$$\frac{}{(t, E[\mathsf{o.f} := \mathsf{v}]) \xrightarrow{write(t,\mathsf{o},\mathsf{f},\mathsf{v})} \{(t, E[\mathbf{()}])\}}\ \textsc{Write}$$

$$\frac{\mathsf{body}(\mathsf{C}, \mathsf{m}) = (\mathsf{T}\ \mathsf{m}(\overline{\mathsf{T}\ \mathsf{x}}) = \mathsf{e})}{(t, E[\mathsf{o.m}(\bar{\mathsf{v}})]) \xrightarrow{type(t,\mathsf{o},\mathsf{C})} \{(t, E[\mathsf{e}[\mathsf{o}, \bar{\mathsf{v}}/\textbf{this}, \bar{\mathsf{x}}]])\}}\ \textsc{Call}$$

$$\frac{\mathsf{body}(\mathsf{C}, \mathsf{m}) = (\textbf{unit}\ \mathsf{m}() = \mathsf{e}) \qquad t \neq t'}{(t, E[\textbf{fork}(\mathsf{o.m})]) \xrightarrow{fork(t,\mathsf{o},\mathsf{C},t')} \{(t, E[\mathbf{()}]), (t', \mathsf{e}[\mathsf{o}/\textbf{this}])\}}\ \textsc{Fork}$$

$$\frac{}{(t, E[\textbf{fence}]) \xrightarrow{fence(t)} \{(t, E[\mathbf{()}])\}}\ \textsc{Fence}$$

$$\frac{}{(t, \mathsf{e}) \xrightarrow{flush(t)} \{(t, \mathsf{e})\}}\ \textsc{Flush}$$

Figure 3: Thread-local semantics — the non-fault cases.

$$\frac{t \in \operatorname{dom} T \qquad (t, T(t)) \xrightarrow{a} T' \qquad \mu' \in [\![a]\!](\mu)}{(\mu, T) \to (\mu', (T - t) + T')}\ \textsc{Step}$$

Figure 4: Single step program evaluation

$$\mathrm{lookup}(-.-,-,-) : \mathit{OId} \times \mathit{FName} \times \mathit{SBuffer} \times \mathit{Heap} \rightharpoonup \mathit{Val}$$

$$\mathrm{lookup}(o.f, \alpha, h) \stackrel{\text{def}}{=} \begin{cases} \bot & \text{when } (o,f) \notin \mathrm{dom}(h) \\ v & \text{when } h(o,f) = v \wedge \forall v' \in \mathit{Val}.\ (o,f,v') \notin \alpha \\ & \quad \text{or } \exists \beta, \gamma \in \mathit{SBuffer}.\ \alpha = \beta \cdot (o,f,v) \cdot \gamma \wedge \\ & \quad\quad \forall v' \in \mathit{Val}.\ (o,f,v') \notin \gamma \end{cases}$$

$$\mathrm{flush}(-,-) : \mathit{Heap} \times \mathit{SBuffer} \to \mathit{Heap}$$

$$\mathrm{flush}(h, \varepsilon) \stackrel{\text{def}}{=} h$$

$$\mathrm{flush}(h, (o,f,v) \cdot \alpha) \stackrel{\text{def}}{=} \begin{cases} \mathrm{flush}(h[(o,f) \mapsto v], \alpha) & \text{when } (o,f) \in \mathrm{dom}(h) \\ \mathrm{flush}(h, \alpha) & \text{otherwise} \end{cases}$$

Figure 5: Generic mathematical functions used throughout the report.

terminal program state that is not a fault is the one, in which all the store buffers are flushed, and all the threads have computed their respective values.

Lemma 1 (Decompose)**.** *For any consistent, non-error program state* $((h, U), T)$ *one of the following holds:*

- *For any thread* $t \in \mathrm{dom}(T)$ *there exists a value* $v \in \mathit{Val}$ *such that* $T(t) = v$ *and* $U(t) = \varepsilon$
- *There exists a (possibly error) state* (μ', T') *such that* $(\mu, T) \to (\mu', T')$*.*

2 The Model

The construction in this section follows closely the one presented in iCAP. As in that case, the non-recursive parts of the model are defined in the category of Sets, **Sets**, while the recursive ones — in the topos of trees, $\mathcal{S}$.

2.1 Local States, Shared States and the Treatment of Store Buffers

Local states consist of a phantom heap, the usual, physical heap and the state transition capability map. The phantom heap is a finite map from objects identifiers and field names to non-zero permissions and *phantom values.* For the latter set, we assume a type PVal that contains the values of the language, Val and is closed under usual mathematical constructions: products, disjoint sums, sequences, etc. This allows us to store, for instance, lists of pairs of values in a phantom field. Shared states consist of an abstract state and a labelled transition system for each shared region. For the buffered updates we use the physical store buffers defined in Section 1.

$$\llbracket \varepsilon \rrbracket(h,U) = \{(h,U)\}$$

$$\llbracket \lightning \rrbracket(h,U) = \{\lightning\}$$

$$\llbracket read(t,o,f,v) \rrbracket(h,U) = \begin{cases} \{(h,U)\} & \text{if } (o,f) \in \operatorname{dom} h \text{ and } \operatorname{lookup}(o.f, U(t), h) = v \\ \emptyset & \text{if } t \notin \operatorname{dom} U \text{ or } (o,f) \in \operatorname{dom} h \text{ and } \operatorname{lookup}(o.f, U(t), h) \neq v \\ \{\lightning\} & \text{if } (o,f) \notin \operatorname{dom} h \end{cases}$$

$$\llbracket write(t,o,f,v) \rrbracket(h,U) = \begin{cases} \{(h, U[t \mapsto U(t) \cdot (o,f,v)])\} & \text{if } (o,f) \in \operatorname{dom} h \text{ and } t \in \operatorname{dom} U \\ \{\lightning\} & \text{if } (o,f) \notin \operatorname{dom} h \\ \emptyset & \text{if } t \notin \operatorname{dom} U \end{cases}$$

$$\llbracket new(t,T,o) \rrbracket(h,U) = \begin{cases} \{(h[(o,\overline{f}) \mapsto \textsf{\textbf{null}}, o \mapsto T], U)\} & \text{if } o \notin \operatorname{dom} h \text{ and } \textsf{fields}(T) = \overline{f} \\ \emptyset & \text{if } o \in \operatorname{dom} h \\ \{\lightning\} & \text{if } \textsf{fields}(T) \text{ undefined} \end{cases}$$

$$\llbracket cas(t,o,f,v_o,v_n,r) \rrbracket(h,U) = \begin{cases} \{(\operatorname{flush}(h, U(t) \cdot (o,f,v_n)), U[t \mapsto \varepsilon])\} & \text{if } (o,f) \in \operatorname{dom} h,\ r = \textsf{\textbf{true}} \\ & \text{and } \operatorname{lookup}(o.f, U(t), h) = v_o \\ \{(\operatorname{flush}(h, U(t)), U[t \mapsto \varepsilon])\} & \text{if } (o,f) \in \operatorname{dom} h,\ r = \textsf{\textbf{false}} \\ & \text{and } \operatorname{lookup}(o.f, U(t), h) \neq v_o \\ \{\lightning\} & \text{if } (o,f) \notin \operatorname{dom} h \\ \emptyset & \text{otherwise} \end{cases}$$

$$\llbracket type(t,o,C) \rrbracket(h,U) = \begin{cases} \{(h,U)\} & \text{if } o \in \operatorname{dom}_t h \text{ and } h_t(o) = C \\ \emptyset & \text{if } o \in \operatorname{dom}_t h \text{ and } h_t(o) \neq C \\ \{\lightning\} & \text{if } o \notin \operatorname{dom}_t h \end{cases}$$

$$\llbracket fork(t,o,C,t') \rrbracket(h,U) = \begin{cases} \{(h, U[t' \mapsto \varepsilon])\} & \text{if } o \in \operatorname{dom}_t h,\ h_t(o) = C \text{ and } t' \notin \operatorname{dom} U \\ \emptyset & \text{if } o \in \operatorname{dom}_t h \text{ and } t' \in \operatorname{dom} U \text{ or } h_t(o) \neq C \\ \{\lightning\} & \text{if } o \notin \operatorname{dom}_t h \end{cases}$$

$$\llbracket fence(t) \rrbracket(h,U) = \begin{cases} \{(\operatorname{flush}(h, U(t)), U[t \mapsto \varepsilon])\} & \text{if } t \in \operatorname{dom} U \\ \emptyset & \text{otherwise} \end{cases}$$

$$\llbracket flush(t) \rrbracket(h,U) = \begin{cases} \{(h[(o,f) \mapsto v], U[t \mapsto \alpha])\} & \text{if } U(t) = (o,f,v) \cdot \alpha \text{ and } (o,f) \in \operatorname{dom} h \\ \emptyset & \text{if } t \notin \operatorname{dom}(U),\ U(t) = \varepsilon, \text{ or } (o,f) \notin \operatorname{dom} h \end{cases}$$

$$\llbracket a \rrbracket(\lightning) = \{\lightning\}$$

Figure 6: Action semantics

$$
\begin{aligned}
Perm &\stackrel{\text{def}}{=} \{q \in \mathbb{Q} \mid 0 \leq q \leq 1\} && \text{(Permissions)}\\
Perm^{+} &\stackrel{\text{def}}{=} Perm \setminus \{0\} && \text{(Non-zero permissions)}\\
Cap &\stackrel{\text{def}}{=} \{f \in (RId \times AId) \to Perm \mid \exists R \subseteq_{fin} RId. && \text{(Capabilities)}\\
&\qquad \forall r \in RId \setminus R.\ \forall \alpha \in AId.\ f(r,\alpha) = 0\}\\
PHeap &\stackrel{\text{def}}{=} OId \times FName \rightharpoonup_{fin} (Perm^{+} \times PVal) && \text{(Phantom heaps)}\\
l \in LState &\stackrel{\text{def}}{=} PHeap \times Heap \times Cap && \text{(Local States)}\\
LTS &\stackrel{\text{def}}{=} AId \to \mathcal{P}(SId \times SId) && \text{(Transition Systems)}\\
s \in SState &\stackrel{\text{def}}{=} RId \rightharpoonup_{fin} (SId \times LTS) && \text{(Shared States)}\\
AState &\stackrel{\text{def}}{=} LState \times SState \times SPool
\end{aligned}
$$

We write $l.p$, $l.h$ and $l.c$ to refer to the appropriate components of the local state l, while $s(r).s$ and $s(r).p$ to refer to the state identifier and transition system components of region r in a shared state s.

Local State Composition

$$
\begin{aligned}
p_1 \cdot p_2 = p_r \iff\ & (\forall (o,f) \in \mathrm{dom}(p_1) \setminus \mathrm{dom}(p_2).\ p_1(o,f) = p_r(o,f)) \land\\
& (\forall (o,f) \in \mathrm{dom}(p_2) \setminus \mathrm{dom}(p_1).\ p_2(o,f) = p_r(o,f)) \land\\
& (\forall (o,f) \in \mathrm{dom}(p_1) \cap \mathrm{dom}(p_2).\ p_1(o,f).c + p_2(o,f).c = p_r(o,f).c \land\\
& \quad p_1(o,f).v = p_2(o,f).v = p_r(o,f).v) \land\\
& \mathrm{dom}(p_r) = \mathrm{dom}(p_1) \cup \mathrm{dom}(p_2)\\
\bullet_{LState} = \{(l_1, l_2, l_r) \mid\ & \mathrm{dom}(l_1.h) \cap \mathrm{dom}(l_2.h) = \emptyset \land\\
& (\forall t \in \Delta(RId \times AId).\ l_1.c(t) + l_2.c(t) \leq 1) \land\\
& l_r.h = l_1.h \cup l_2.h \land\ l_r.c = l_1.c + l_2.c \land l_r.p = l_1.p \cdot l_2.p\},
\end{aligned}
$$

where $+$ on finite maps denotes pointwise addition. The composition of phantom heaps is defined when all the elements that are in the domains of both heaps are mapped to the same values with permissions that are jointly not greater then 1.

The relation is functional, so it induces a partial function

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

Lemma 2. $\forall l_1, l_2, l_3 \in \Delta(LState).\ \triangleright (l_1 = l_2 \bullet_{LState} l_3) \Rightarrow (l_1 = l_2 \bullet_{LState} l_3 \lor \triangleright\bot)$

Proof. Using the theory of upwards-closure from the iCAP technical report [6] □

Action Allowed

$$
\begin{aligned}
\mathrm{act}_\circ &: LState \times RId \to \mathcal{P}(AId) \in \mathbf{Sets}\\
\mathrm{act}_\circ(l,r) &\stackrel{\text{def}}{=} \{\alpha \in AId \mid l.c(r,\alpha) < 1\}
\end{aligned}
$$

Update Allowed

$$\mathrm{upd}_\circ : LState \times RId \times LTS \to \mathcal{P}(SId \times SId) \in \mathbf{Sets}$$
$$\mathrm{upd}_\circ(l, r, p) \stackrel{\mathrm{def}}{=} \{(s_1, s_2) \in SId \times SId \mid \exists \alpha \in \mathrm{act}_\circ(l, r).\ (s_1, s_2) \in p(\alpha)\}$$

2.2 Interference Relation

In our setup, interference is more complicated then in plain iCAP, due to the existence of store buffered updates. While in iCAP the interference concerned only the shared regions and transitions they are allowed to make, we also have to consider the three kinds of actions the environment can take with respect to the buffers: the allocation of a new store buffer, a buffered update written into an existing one, or a flushing of an update from store buffer to the heap. The interference is thus defined as follows.

$$\begin{aligned}
(l, U_1)\ R_n\ (l, U_2) \text{ iff } & \mathrm{dom}(U_1) \subseteq \mathrm{dom}(U_2) \wedge \forall t \in \mathrm{dom}(U_2) \setminus \mathrm{dom}(U_1).\ U_2(t) = \varepsilon \wedge \\
& \forall t \in \mathrm{dom}(U_1).\ U_2(t) = U_1(t) \\
(l, U_1)\ R_w\ (l, U_2) \text{ iff } & \mathrm{dom}(U_1) = \mathrm{dom}(U_2) \wedge \\
& \exists t \in \mathrm{dom}(U_2), o \in OId, f \in FName, v \in Val.\ (o, f) \notin \mathrm{dom}(l.h) \wedge \\
& \ U_2(t) = U_1(t) \cdot (o, f, v) \wedge \forall t' \in \mathrm{dom}(U_2).\ t \neq t' \Rightarrow U_1(t) = U_2(t) \\
(l_1, U_1)\ R_f\ (l_2, U_2) \text{ iff } & \mathrm{dom}(U_1) = \mathrm{dom}(U_2) \wedge \\
& \exists t \in \mathrm{dom}(U_1), o \in OId, f \in FName, v \in Val.\ U_1(t) = (o, f, v) \cdot U_2(t) \\
& \ l_2 = \mathrm{flush}(l_1, (o, f, v)) \wedge \forall t' \in \mathrm{dom}(U_2).\ t \neq t' \Rightarrow U_1(t) = U_2(t) \\
R_{sb} & \stackrel{\mathrm{def}}{=} (R_n \cup R_w \cup R_f)^* \\
(l_1, s_1)\ R_r^A\ (l_2, s_2) \text{ iff } & l_1 \leq l_2 \wedge \mathrm{dom}(s_1) \subseteq \mathrm{dom}(s_2) \\
& \forall r \in \mathrm{dom}(s_1).\ (s_1(r).s, s_2(r).s) \in (\bigcup_{\alpha \in A, l_1.c(r,\alpha) < 1} S(r).t(\alpha))^* \wedge s_1(r).p = s_2(r).p \\
(l_1, s_1, U_1)\ R'_A(l_2, s_2, U_2) \text{ iff } & ((l_1, s_1)\ R_r^A\ (l_2, s_2) \wedge U_1 = U_2) \vee \\
& ((l_1, U_1)\ R_{sb}\ (l_2, U_2) \wedge (s_1 \leq s_2)) \\
R_A & \stackrel{\mathrm{def}}{=} (R'_A)^*
\end{aligned}$$

In the above R^* denotes the reflexive, transitive closure of R. We use R as a shorthand for R_{AId}.

2.3 Region Interpretations

We follow the iCAP treatment to give the concrete interpretation to the abstract shared states in the shared regions — and thus define the type of region interpretations as a guarded recursive type in the topos of trees, $\mathcal{S}$. We take $RIntr$ to be an object in $\mathcal{S}$ that satisfies the isomorphism

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

where $\Delta(AState)$ is upwards-closed with respect to the interference relation R, and the ordering on action models, $AMod \stackrel{\text{def}}{=} \Delta(RId) \rightharpoonup_{fin} RIntr$, is given as

$$\zeta_1 \leq \zeta_2 \stackrel{\text{def}}{=} \text{dom}(\zeta_1) \subseteq \text{dom}(\zeta_2) \wedge \forall r \in \text{dom}(\zeta_1).\ \zeta_1(r) = \zeta_2(r).$$

The ordering on $\Delta(SId)$ is discrete.

Now we can define the maps between $\Delta(SId) \times AMod \rightarrow_{mon} \mathcal{P}^{\uparrow}(\Delta(AState))$ and $RIntr$ as follows:

$$\begin{aligned}
&lam : (\Delta(SId) \times AMod \rightarrow_{mon} \mathcal{P}^{\uparrow}(\Delta(AState))) \rightarrow RIntr \\
&lam \stackrel{\text{def}}{=} i^{-1} \circ next \\
&app : RIntr \rightarrow (\Delta(SId) \times AMod \rightarrow_{mon} \mathcal{P}^{\uparrow}(\Delta(AState))) \\
&app \stackrel{\text{def}}{=} \lambda x : RIntr.\ \lambda(s,\zeta) : \Delta(SId) \times (AMod).\ \lambda a : \Delta(AState). \\
&\qquad succ(J(J(i(x))(next(s), next(\zeta)))(next(a)))
\end{aligned}$$

Lemma 3. $app \circ lam = \triangleright$

2.4 Instrumented States

Instrumented states consist of a concrete local state and store buffer pool, an abstract state transition system for each shared region, and concrete interpretation of each abstract shared state:

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

We use $m.l$, $m.s$, $m.U$ and $m.a$ to refer to the appropriate components of the instrumented state $m : \mathcal{M}$.

We also define a subset of $\mathcal{M}$ that omits the store buffers, which we will sometimes use:

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

We use analogous projections whenever needed.

Propositions We define two spaces of propositions, $Prop_{TSO}$, which denotes the general assertions, and $Prop_{SC}$, for assertions that do not mention the store buffers.

$$\begin{aligned}
Prop_{TSO} &\stackrel{\text{def}}{=} \mathcal{P}^{\uparrow}(\mathcal{M}) \\
Prop_{SC} &\stackrel{\text{def}}{=} \mathcal{P}^{\uparrow}(\mathcal{H}),
\end{aligned}$$

where the orderings are pointwise extension orderings.

We define an interpretation of $Prop_{SC}$ in $Prop_{TSO}$ as the propositions that have a "locally flushed" subset:

$$\begin{aligned}
&\text{lfd} \subseteq LState \times SPool \in \textbf{Sets} \\
&\text{lfd}(l,U) \text{ iff } \forall t \in TId.\ \forall (o,f) \in \text{dom}(l).\ \forall v \in Val.\ (o,f,v) \notin U(t) \\
&\ulcorner - \urcorner : Prop_{SC} \rightarrow Prop_{TSO} \in \mathcal{S} \\
&\ulcorner p \urcorner \stackrel{\text{def}}{=} \{(l,s,U,\zeta) \in \mathcal{M} \mid \exists l' \leq l.\ (l',s,\zeta) \in p \wedge \text{lfd}(l',U)\}
\end{aligned}$$

Interpreted in this way $Prop_{SC}$ defines a subset of well-behaved propositions that do not depend on the state of the buffers.

Erasure

$$\lfloor (s,\zeta) \rfloor_r \stackrel{\text{def}}{=} \{(l,U) \in LState \times SPool \mid (l,s,U) \in app(\zeta(r))(s(r).s,\zeta)\}$$
$$\begin{aligned}\lfloor (l,s,U,\zeta) \rfloor_A \stackrel{\text{def}}{=} \{(h,U) \in Heap \times SPool \mid & \\ & \exists l' \in LState, sr \in \mathrm{dom}(s) \cap A \to LState. \\ & \quad h = l'.h \wedge l' = l \bullet \Pi_{r \in \mathrm{dom}(s) \cap A} sr(r) \wedge \\ & \quad (\forall (o,f) \in \mathrm{dom}(l'.p).\ \exists f' \in FName.\ (o,f') \in \mathrm{dom}(l'.h)) \wedge \\ & \quad \forall r \in \mathrm{dom}(s) \cap A.\ (sr(r),U) \in \lfloor (s,\zeta) \rfloor_r\}\end{aligned}$$

As usual, we use $\lfloor m \rfloor$ as a shorthand for $\lfloor m \rfloor_{RId}$.

Composition

$$\bullet_{\mathcal{M}} \stackrel{\text{def}}{=} \bullet_{LState} \times \bullet_{=} \times \bullet_{=} \times \bullet_{=}$$

2.5 Logical Connectives

Given below are the definitions for connectives in $Prop_{TSO}$. The definitions for $Prop_{SC}$ are analogous.

$$\begin{aligned} emp &\stackrel{\text{def}}{=} \mathcal{M} \\ p * q &\stackrel{\text{def}}{=} \{m \in \mathcal{M} \mid \exists m_1, m_2 \in \mathcal{M}.\ m = m_1 \bullet m_2 \wedge m_1 \in p \wedge m_2 \in q\} \\ p \Rightarrow q &\stackrel{\text{def}}{=} \{m \in \mathcal{M} \mid \forall m' \geq m.\ m' \in p \Rightarrow m' \in q\} \\ \exists_\tau(p) &\stackrel{\text{def}}{=} \{m \in \mathcal{M} \mid \exists x : \tau.\ m \in p(x)\} \\ \forall_\tau(p) &\stackrel{\text{def}}{=} \{m \in \mathcal{M} \mid \forall x : \tau.\ m \in p(x)\} \end{aligned}$$

Points-to predicates We define two points-to predicates, one for the physical heap and one for the phantom heap. Both of these are in $Prop_{SC}$. Since the store buffers do not influence the phantom state, we will freely omit the embeddings over it.

$$\begin{aligned} o.f \mapsto v &\stackrel{\text{def}}{=} \{(l,s,\zeta) \in \mathcal{H} \mid l.h(o,f) = v\} \\ o_f \stackrel{\pi}{\mapsto} v &\stackrel{\text{def}}{=} \{(l,s,\zeta) \in \mathcal{H} \mid \exists \pi' \in Perm^+.\ l.p(o,f) = (\pi',v) \wedge \pi' \geq \pi\} \end{aligned}$$

Lemma 4. *Let X be a total and inhabited object in $\mathcal{S}$. Then*

$$\forall p \in X \to Prop_{TSO}.\ \triangleright \exists_X(p) \iff \exists_X(\lambda x. \triangleright p(x))$$

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

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

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

Likewise, let $m \in \mathcal{M}$ such that $m \in \exists_X(\lambda x. \triangleright p(x))$ then $\exists x : X.\ \triangleright(m \in p(x))$ and thus $\triangleright(\exists x : X.\ m \in p(x))$. □

Properties of the Embedding In order to use the $\ulcorner - \urcorner$ embedding to define the two separation logics, we need it to satisfy certain properties. These are formalized in the next few lemmas.

Lemma 5. *Embedding preserves the entailment relation, i.e., for any $p, q \in \mathit{Prop}_{SC}$,*

$$(p \subseteq q) \iff (\ulcorner p \urcorner \subseteq \ulcorner q \urcorner).$$

Proof ($\Rightarrow$). Assume $p \subseteq q$, and take any $(l, s, U, \zeta) \in \ulcorner p \urcorner$. By definition, this gives us $l' \leq l$, such that $(l', s, \zeta) \in p$ and $\mathrm{lfd}(l', U)$. By assumption, we have $(l', s, \zeta) \in q$, so we can take l' as a witness, and the properties hold trivially. □

Proof ($\Leftarrow$). Assume $\ulcorner p \urcorner \subseteq \ulcorner q \urcorner$, and take $(l, s, \zeta) \in p$. This means $(l, s, \emptyset, \zeta) \in \ulcorner p \urcorner$, since $\mathrm{lfd}(l, \emptyset)$ holds trivially, and so we get $(l, s, \emptyset, \zeta) \in \ulcorner q \urcorner$. From this, we obtain $l' \leq l$ such that $(l', s, \zeta) \in q$, but since q is upwards closed, $(l, s, \zeta) \in q$ holds. □

Lemma 6. *For any $p \in \mathit{Prop}_{TSO}$ and any $m \in p$ there exists a local state $l' \leq m.l$ such that $\mathrm{lfd}(l', m.U)$ and l' is the maximal state with this property (wrt the extension ordering).*

Proof. Take any $p \in \mathit{Prop}_{TSO}$ and any $(l, s, U, \zeta) \in p$. Let s be the set of all the locations in the store-buffer pool U: formally,

$$(o, f) \in s \text{ iff } \exists t \in \mathrm{dom}(U).\ \exists v \in \mathit{Val}.\ (o, f, v) \in U(t).$$

Take the state $l' = l\ |_{\mathrm{dom}(l) \setminus s}$ as the witness. Both $l' \leq l$ and $\mathrm{lfd}(l', U)$ clearly hold, so it suffices to show that this is the biggest state with this property. To this end, take any $l'' \leq l'$ such that $\mathrm{lfd}(l'', U)$. We need to show that $l'' \leq l'$. To this end, take any $(o, f) \in \mathrm{dom}(l'')$. Since both local states are smaller than l under extension ordering, it suffices to show that $(o, f) \in \mathrm{dom}(l')$ – the values (o, f) gets mapped to will have to agree. Clearly, we have $(o, f) \in \mathrm{dom}(l)$, since $l'' \leq l$, so it suffices to show that $(o, f) \notin s$. To this end, assume $(o, f) \in s$. This gives us $t \in \mathrm{dom}(U)$ and $v \in \mathit{Val}$ such that $(o, f, v) \in U(t)$. However, since $\mathrm{lfd}(l'', U)$, we know that $(o, f, v) \notin U(t)$, which ends the proof. □

Lemma 7. *Embedding preserves limits, i.e., for any τ and $p : \tau \to \mathit{Prop}_{SC}$,*

$$\forall_\tau(\ulcorner p \urcorner) = \ulcorner \forall_\tau(p) \urcorner.$$

Proof (⊆). Take any $(l, s, U, \zeta) \in \forall_\tau(\ulcorner p \urcorner)$. By definition of universal quantifier, this means that $\forall x : \tau.\ (l, s, U, \zeta) \in \ulcorner p(x) \urcorner$. By Lemma 6, we get l', the greatest local state such that $l' \leq m.l$ and $\text{lfd}(l', U)$. Now it suffices to show that $(l', s, \zeta) \in \forall_\tau(p)$. Take any $x : \tau$. From the assumption, we get $l'' \leq l$ such that $\text{lfd}(l'', U)$ and $(l'', s, \zeta) \in p(x)$. However by the universal property of l', we know that $l'' \leq l'$, and so, by upwards closure $(l', s, \zeta) \in p$. □

Proof (⊇). Take any $(l, s, U, \zeta) \in \ulcorner \forall_\tau(p) \urcorner$. By definition, this gives us $l' \leq l$ such that $\text{lfd}(l', U)$ and $\forall x : \tau.\ (l', s, \zeta) \in p(x)$. Taking any $x : \tau$, and picking l' as the witness trivially finishes the proof. □

Lemma 8. *Embedding preserves colimits, i.e., for any τ and $p : \tau \to Prop_{SC}$,*

$$\exists_\tau(\ulcorner p \urcorner) = \ulcorner \exists_\tau(p) \urcorner.$$

Proof (⊆). Take any $(l, s, U, \zeta) \in \exists_\tau(\ulcorner p \urcorner)$. Unrolling the definitions, this gives us $x : \tau$ and $l' \leq l$ such that $\text{lfd}(l', U)$ and $(l', s, \zeta) \in p(x)$. Taking l' and x as witnesses provides all the properties we need to finish the proof. □

Proof (⊇). Take any $(l, s, U, \zeta) \in \ulcorner \exists_\tau(p) \urcorner$. Unrolling the definitions gives us $l' \leq l$ such that $\text{lfd}(l', U)$ and $x : \tau$ such that $(l', s, \zeta) \in p(x)$. Taking x and l' as witnesses provides all the needed properties. □

Lemma 9. *Embedding preserves separating conjunctions, i.e., for any $p, q \in Prop_{SC}$,*

$$\ulcorner p \urcorner * \ulcorner q \urcorner = \ulcorner p * q \urcorner.$$

Proof (⊆). Take any $(l, s, U, \zeta) \in \ulcorner p \urcorner * \ulcorner q \urcorner$. This means there exist $l_1, l_2 \in \Delta(LState)$ such that $l_1 \bullet l_2 = l$, $(l_1, s, U, \zeta) \in \ulcorner p \urcorner$ and $(l_2, s, U, \zeta) \in \ulcorner q \urcorner$. By definition of the embedding, this gets us $l_1' \leq l_1$ and $l_2' \leq l_2$ such that $(l_1', s, \zeta) \in p$, $(l_2', s, \zeta) \in q$, $\text{lfd}(l_1', U)$ and $\text{lfd}(l_2', U)$. Pick $l_1' \bullet l_2' \leq l$ as a witness. Obviously, $\text{lfd}(l_1' \bullet l_2', U)$, and we can split the state as l_1' and l_2' to show that $(l_1', s, \zeta) \in p$ and $(l_2', s, \zeta) \in q$. □

Proof (⊇). Take any $(l, s, U, \zeta) \in \ulcorner p * q \urcorner$. This means that there exists $l' \leq l$, l_1 and l_2 such that $\text{lfd}(l', U)$, $l' = l_1 \bullet l_2$, $(l_1, s, \zeta) \in p$ and $(l_2, s, \zeta) \in q$. Pick as witnesses l_1 and $l'' = l \mid_{\text{dom}(l) \setminus \text{dom}(l_1)}$. These states compose to give l, so we need to show $(l_1, s, U, \zeta) \in \ulcorner p \urcorner$ and $(l'', s, U, \zeta) \in \ulcorner q \urcorner$. For the first of these, pick l_1 as the witness. Since $l_1 \leq l'$, we have $\text{lfd}(l_1, U)$, and we have $(l_1, s, \zeta) \in p$ by assumption. For the second goal, pick l_2 as the witness. Since l_2 is compatible with l_1, we get $l_2 \leq l''$, and since $l_2 \leq l'$ — $\text{lfd}(l_2, U)$. By assumption we also have $(l_2, s, \zeta) \in q$. □

Validity

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

Region Assertion and Interpretation

$$region : \mathcal{P}(\Delta(SId)) \times \Delta(LTS) \times \Delta(RId) \to Prop_{SC} \in \mathcal{S}$$
$$region(X, p, r) \stackrel{\text{def}}{=} \{(l, s, \zeta) \in \mathcal{H} \mid s(r).s \in X \wedge s(r).p = p\}$$
$$rintr : (\Delta(SId) \to Prop_{TSO}) \times \Delta(RId) \to Prop_{SC} \in \mathcal{S}$$
$$rintr(I, r) \stackrel{\text{def}}{=} \{(l, s, \zeta) \in \mathcal{H} \mid r \in \text{dom}(\zeta) \wedge \forall x \in \Delta(SId), \zeta' \geq \zeta. \; app(\zeta(r))(x, \zeta') = \triangleright(\lambda(l, s, U).I(x)(l, s, U, \zeta'))\}$$

Note that although technically these assertions are in $Prop_{SC}$, we will often omit the $\ulcorner - \urcorner$ interpretation when using them, since they do not depend at all on the local state, so one can always pick the empty local state as the appropriate witness.

Action Permission

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

2.6 Stability

$$\underset{A}{\text{stable}}(p) \stackrel{\text{def}}{=} R_A(p) \subseteq p$$
$$\text{bstable}(p) \stackrel{\text{def}}{=} R_{sb}(p) \subseteq p$$
$$\underset{A}{\text{rstable}}(p) \stackrel{\text{def}}{=} R^A_r(p) \subseteq p$$

As usual, we use stable as a shorthand for stable_{RId}

Lemma 10. $\forall p \in Prop_{TSO}.\ \text{stable}(p) \iff \text{bstable}(p) \wedge rstable(p)$

Lemma 11. $R(emp) \subseteq emp \wedge \forall p, q \in Prop_{TSO}.\ R(p * q) \subseteq R(p) * R(q)$

Lemma 12. $\forall A \in \mathcal{P}(\Delta(RId)), p \in Prop_{TSO}.\ \text{stable}_A(p) \Rightarrow \text{stable}_A(\triangleright p)$

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

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

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

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

and so $m_2 \in \triangleright p$. □

2.7 View Shift

$$p \sqsubseteq_A q \stackrel{\text{def}}{=} \forall r \in Prop_{TSO}.\ \underset{A}{\text{stable}}(r) \Rightarrow \lfloor p * r \rfloor_A \subseteq \lfloor q * r \rfloor_A$$

Again, we write $\sqsubseteq$ as a shorthand for $\sqsubseteq_{RId}$.

Lemma 13. $\forall p, q, r \in Prop_{TSO}.\ \text{stable}(r) \wedge p \sqsubseteq_A q \Rightarrow p * r \sqsubseteq_A q * r$

Lemma 14. $\forall p_1, p_2, q_1, q_2 \in Prop_{TSO}.\ p_1 \sqsubseteq q_1 \wedge p_2 \sqsubseteq q_2 \Rightarrow p_1 * p_2 \sqsubseteq q_1 * q_2$

2.8 Atomic Satisfaction

$$\begin{aligned} a \text{ sat}_A\ \{p\}\,\{q\} \stackrel{\text{def}}{=} {} & \forall r \in \mathit{Prop}_{TSO}, m \in \mathcal{M}, h \in \mathit{Heap}, U \in \mathit{SPool}. \\ & m \in p * \triangleright r \wedge (h, U) \in \lfloor m \rfloor_A \wedge \underset{A}{\text{stable}}(r) \Rightarrow \\ & (\triangleright \lightning \notin [\![a]\!](h, U)) \wedge \\ & \forall (h', U') \in [\![a]\!](h, U).\ \exists m' \in \mathcal{M}.\ \triangleright (m' \in q * r \wedge (h', U') \in \lfloor m' \rfloor_A) \end{aligned}$$

We write a sat $\{p\}\,\{q\}$ for a sat$_{RId}$ $\{p\}\,\{q\}$, as usual.

Lemma 15.

$$\forall A \in \mathcal{P}(\Delta(\mathit{RId})).\ \forall a \in \Delta(\mathit{Act}).\ \forall p, q \in \mathit{Prop}_{TSO}.\ \triangleright \bot \Rightarrow a \text{ sat}_A\ \{p\}\,\{q\}$$

Proof. Follows easily as $\triangleright\bot$ implies $\triangleright\lightning \notin [\![a]\!](h, U)$ and $\triangleright(m' \in q * r \wedge (h', U') \in \lfloor m' \rfloor_A)$ for any m', h' and U', and $\mathcal{M}$ is inhabited. □

Lemma 16 (Atomic-Shift)**.**

$$\begin{aligned} & \forall A \in \mathcal{P}(\Delta(\mathit{RId})), p_1, p_2, q_1, q_2 \in \mathit{Prop}_{TSO}. \\ & \quad p_1 \sqsubseteq_A p_2 \wedge a \text{ sat}_A\ \{p_2\}\,\{q_2\} \wedge \triangleright(q_2 \sqsubseteq_A q_1) \Rightarrow a \text{ sat}_A\ \{p_1\}\,\{q_1\} \end{aligned}$$

Proof. Assume $r \in \mathit{Prop}_{TSO}$, $m \in \mathcal{M}$, $h, h' \in \mathit{Heap}$, and $U, U' \in \mathit{SPool}$ such that

$$\underset{A}{\text{stable}}(r) \qquad m \in p_1 * \triangleright r \qquad (h, U) \in \lfloor m \rfloor_A \qquad (h', U') \in [\![a]\!](h, U).$$

By Lemma 12 it follows that stable$_A(\triangleright r)$ and thus $(h, U) \in \lfloor p * \triangleright r \rfloor_A \subseteq \lfloor q * \triangleright r \rfloor_A$. Hence, there exists $m' \in \mathcal{M}$ such that $\triangleright(m' \in q_2 * r \wedge (h', U') \in \lfloor m' \rfloor_A)$, and so

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

which ends the proof. □

Lemma 17 (Atomic-Frame)**.**

$$\begin{aligned} & \forall A \in \mathcal{P}(\Delta(\mathit{RId})), p, q, r \in \mathit{Prop}_{TSO}. \\ & \quad \underset{A}{\text{stable}}(r) \wedge a \text{ sat}_A\ \{p\}\,\{q\} \Rightarrow a \text{ sat}_A\ \{p * \triangleright r\}\,\{q * r\} \end{aligned}$$

Proof. Assume $r' \in \mathit{Prop}_{TSO}$, $m \in \mathcal{M}$, $h, h' \in \mathit{Heap}$ and $U, U' \in \mathit{SPool}$ such that

$$\underset{A}{\text{stable}}(r') \qquad m \in p * \triangleright r * \triangleright r' \qquad (h, U) \in \lfloor m \rfloor_A \qquad (h', U') \in [\![a]\!](h, U).$$

This means that $m \in p * \triangleright(r * r')$. Since stable$_A(r * r')$, from definition of atomic satisfiability it follows that there exists an $m' \in \mathcal{M}$ such that

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

which ends the proof. □

Corollary 1.

$$\forall A \in \mathcal{P}(\Delta(RId)), p, q, r \in Prop_{TSO}.$$
$$\underset{A}{\text{stable}}(r) \wedge a \text{ sat}_A \ \{p\}\{q\} \Rightarrow a \text{ sat}_A \ \{p * r\}\{q * r\}$$

Lemma 18 (Id-Refl)**.** $\forall A \in \mathcal{P}(\Delta(RId)), p \in Prop_{TSO}.\ \varepsilon \text{ sat}_A \ \{p\}\{p\}$

Proof. Take a $r \in Prop_{TSO}$, $m \in \mathcal{M}$, $h \in Heap$ and $U \in SPool$ such that

$$\underset{A}{\text{stable}}(r) \qquad m \in p * \triangleright r \qquad (h, U) \in \lfloor m \rfloor_A$$

Since $[\![\varepsilon]\!](h, U) = \{(h, U)\}$, so clearly $\lightning \notin [\![\varepsilon]\!](h, U)$. Now, it suffices to exhibit m' such that $(h, U) \in \lfloor m' \rfloor_A$ and $\triangleright(m' \in p * r)$, which follow simply by monotnicity of $\triangleright$ if we choose $m' = m$. ☐

Lemma 19 (Flush-Refl)**.** *For any $A \in \mathcal{P}(\Delta(RId))$, $p \in Prop_{TSO}$, $t \in TId$,*

$$\underset{A}{\text{stable}}(p) \Rightarrow \mathit{flush}(t) \text{ sat}_A \ \{p\}\{p\}.$$

Proof. Take an $r \in Prop_{TSO}$, $m \in \mathcal{M}$, $h \in Heap$ and $U \in SPool$ such that

$$\underset{A}{\text{stable}}(r) \qquad m \in p * \triangleright r \qquad (h, U) \in \lfloor m \rfloor_A.$$

Since $flush$ never faults, we only need to consider the postcondition branch. Take any $(h', U') \in [\![\mathit{flush}(t)]\!](h, U)$. By definition this means that there exist $(o, f) \in \text{dom}(h)$ and v such that $h' = h[(o, f) \mapsto v]$, $U(t) = (o, f, v) \cdot U'(t)$ and $U(t') = U'(t')$ for any $t' \neq t$. As a witness we give $m' = (\text{flush}(m.l, (o, f, v)), m.s, U', m.\zeta)$. Since $(m.l, U)\ R_f\ (m'.l, U')$ and $\text{stable}_A(p * \triangleright r)$, we easily have $\triangleright(m' \in p * r)$ by monotonicity of $\triangleright$. This means we only need to show that $(h', U') \in \lfloor m' \rfloor_A$. From $(h, U) \in \lfloor m \rfloor_A$ we get l_h and sr such that $h = l_h.h$, $l_h = m.l \bullet \Pi_{r \in \text{dom}(m.s) \cap A} sr(r)$ and $(sr(r), U) \in \lfloor (m.s, m.\zeta) \rfloor_r$ for any $r \in \text{dom}(m.s) \cap A$. Take $l'_h = \text{flush}(l_h, (o, f, v))$ and $sr'(r) = \text{flush}(sr(r), (o, f, v))$ as the witnesses. Since flush commutes over composition, we get

$$\begin{aligned} \text{flush}(l_h, (o, f, v)) &= \text{flush}(m.l \bullet \Pi_{r \in \text{dom}(m.s) \cap A} sr(r), (o, f, v)) \\ &= m.l' \bullet \Pi_{r \in \text{dom}(m.s) \cap A} sr'(r). \end{aligned}$$

Since, for every $r \in \text{dom}(m.s) \cap A$ we also have $(sr(r), U)\ R_f\ (sr'(r), U')$ and the erasures of shared regions are closed under interference (since they are defined by the application of an action model), we get $(sr'(r), U') \in \lfloor (m.s, m.\zeta) \rfloor_r$ for each $r \in \text{dom}(m.s) \cap A$, which ends the proof. ☐

2.9 Specification Embedding

$$spec : \Omega \rightarrow Prop_{TSO} \in \mathcal{S}$$
$$spec(s) \stackrel{\text{def}}{=} \{m \in \mathcal{M} \mid s\}$$

Lemma 20 (Sat-Spec-Later)**.**

$$\forall a \in \mathit{Act}, p, q \in \mathit{Prop}_{TSO}, s \in \Omega.$$
$$a \text{ sat } \{p * \mathit{spec}(\triangleright s)\}\,\{q\} \Rightarrow a \text{ sat } \{p * \mathit{spec}(\triangleright s)\}\,\{q * \mathit{spec}(s)\}$$

Proof. Take $r \in \mathit{Prop}_{TSO}$, $m \in \mathcal{M}$, $h, h' \in \mathit{Heap}$ and $U, U' \in \mathit{SPool}$ such that

$$\underset{A}{\text{stable}}(r) \qquad m \in p * \mathit{spec}(\triangleright s) * \triangleright r \qquad (h, U) \in \lfloor m \rfloor_A \qquad (h', U') \in [\![a]\!](h, U).$$

By our assumption, there exists an $m' \in \mathcal{M}$, such that $\triangleright(m' \in q * r)$ and $\triangleright(h' \in \lfloor m' \rfloor)$. Hence, $\triangleright((m' \in q * r) \wedge s)$, and so $\triangleright(m' \in q * \mathit{spec}(s) * r)$. □

Lemma 21 (Later-Star)**.** $\triangleright(p * q) = (\triangleright p) * (\triangleright q)$

Proof ($\subseteq$). Assume $m \in \triangleright(p * q)$, and so $\triangleright(m \in p * q)$, and, unfolding the definition of $*$,

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

Now, unfolding the definition of $\bullet$, we get

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

This means we can take the witnesses l_1 and l_2, such that

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

From the first of these properties, by Lemma 2 we get $m.l = l_1 \bullet_{\mathit{LState}} l_2 \vee \triangleright\bot$. We inspect the two cases.

Assume, first, that $m.l = l_1 \bullet_{\mathit{LState}} l_2$. Then, $m = (l_1, m.s, m.U, m.a) \bullet (l_2, m.s, m.U, m.a)$, and so $m \in (\triangleright p) * (\triangleright q)$.

In the other case, assume $\triangleright\bot$. Now, take the splitting of $m = m \bullet (\varepsilon, m.s, m.U, m.a)$. Since $\triangleright\bot$, $\triangleright((\varepsilon, m.s, m.U, m.a) \in q)$, and so $m \in (\triangleright p) * (\triangleright q)$. □

Proof ($\supseteq$). Follows by monotonicity of $\triangleright$. □

2.10 Thread Safety

We define safety by guarded recursion:

$$\text{safe} : \mathcal{P}((\mathit{TId} \times \mathit{Expr}) \times \mathit{Prop}_{TSO} \times (\mathit{Val} \to \mathit{Prop}_{TSO})) \in \mathcal{S}$$
$$\text{safe} \stackrel{\text{def}}{=} \text{fix}\, f.\ \widehat{\text{safe}}(f),$$

where

$$
\begin{aligned}
\widehat{\text{safe}}(f)((t,e),p,q) \overset{\text{def}}{=} & (\exists v \in \mathit{Val}.\ e = v \land p \sqsubseteq q(v)) \lor \\
& \forall T \in \Delta(\mathit{TPool}), a \in \Delta(\mathit{Act}).\ (t,e) \xrightarrow{a} T \Rightarrow \\
& \quad \exists p' : \text{dom}(T) \to \mathit{Prop}_{TSO}.\ t \in \text{dom}(T) \land \\
& \quad\quad (\forall t \in \text{dom}(T).\ \triangleright \text{stable}(p'(t))) \land \\
& \quad\quad a \text{ sat } \{p\} \left\{ \circledast_{t \in \text{dom}(T)} p'(t) \right\} \land \\
& \quad\quad \triangleright f((t, T(t)), p'(t), q) \land \\
& \quad\quad \triangleright \forall t' \in \text{dom}(T - t).\ f(t', p'(t'), \lambda v.\ \top)
\end{aligned}
$$

Lemma 22 (Stable-Closed)**.**

$$
\begin{aligned}
& \forall X \in \mathcal{P}(\Delta(\mathit{SId})), p \in \Delta(\mathit{LTS}), r \in \Delta(\mathit{RId}). \\
& \quad (\forall \alpha \in \mathit{AId}.\ \alpha \neq \alpha_i \Rightarrow p(\alpha)(X) \subseteq X) \Rightarrow \text{stable}(\mathit{region}(X,p,r) * \circledast_i [\alpha_i]_1^r)
\end{aligned}
$$

Lemma 23 (ViewShift-Weaken)**.**

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

Lemma 24 (ViewShift-Trans)**.**

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

Lemma 25 (RegInterp-WF)**.**

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

Proof (Monotonicity). Take $x \in \Delta(\mathit{SId})$, $\zeta_1, \zeta_2 \in \mathit{AMod}$ such that $\zeta_1 \leq \zeta_2$. By upwards-closure of p, for any $l \in \mathit{LState}$, $s \in \mathit{SState}$ and $U \in \mathit{SPool}$ such that $(l,s,U,\zeta_1) \in p(x)$, $(l,s,U,\zeta_2) \in p(x)$ also holds. □

Proof (Upwards-closure). Take $x \in \Delta(\mathit{SId})$, $\zeta \in \mathit{AMod}$, $l_1, l_2 \in \mathit{LState}$, $s_1, s_2 \in \mathit{SState}$ and $U_1, U_2 \in \mathit{SPool}$ such that $(l_1, s_1, U_1, \zeta) \in p(x)$ and $(l_1, s_1, U_1)\ R\ (l_2, s_2, U_2)$. By stability of p it follows that $(l_2, s_2, U_2, \zeta) \in p(x)$. □

Lemma 26 (Safe-Sat)**.** *For any thread $t \in \Delta(\mathit{TId})$, closed expression $e \in \Delta(\mathit{Expr})$, $p \in \mathit{Prop}_{TSO}$ and $q \in \Delta(\mathit{Val}) \to \mathit{Prop}_{TSO}$ we have*

$$
\begin{aligned}
& (\forall a \in \Delta(\mathit{Act}).\ \forall v \in \Delta(\mathit{Val}).\ (t,e) \xrightarrow{a} \{(t,v)\} \Rightarrow a \text{ sat } \{p\}\{q(v)\}) \land \\
& \quad \text{stable}(p) \land (\forall v \in \Delta(\mathit{Val}).\ \text{stable}(q(v))) \land \text{atomic}(e) \\
& \quad\quad \Rightarrow \text{safe}((t,e),p,q).
\end{aligned}
$$

Proof. The proof proceeds by Löb induction, which we will need to handle the flushing of the store buffer that can occur before the atomic command is actually executed.

Unfolding the definition of safety, we find that, since e is atomic, we should take the "progress" case. Take any $T \in \mathit{TPool}$ and $a \in \mathit{Act}$ such that $(t, e) \xrightarrow{a} T$. Since e is atomic, we only need to consider the cases when a is a *read*, *write*, *cas* or *flush* action. We proceed by case analysis.

First, let us consider the flushing. By definition of the semantics, we know that $T = \{(t, e)\}$. We take the map p' to be the map given by $t \mapsto p$. Thus, stability of p' follows by stability of p, atomic satisfaction of the $\mathit{flush}(t)$ action – by Lemma 19, and since we did not fork any threads we only need to show that $\triangleright \text{safe}((t, e), p, q)$ – which follows from our Löb inductive hypothesis.

In the other three cases, we have to consider proper reductions. In all of these cases T has the same shape: we have $T = \{(t, v)\}$ for some value v. Thus, we can take the map p' to be given by $t \mapsto q(v)$. Then, the stability of p' follows by stability of q, and the atomic satisfaction by the lemma's assumption. Since we do not allocate any new threads, this only leaves us with showing that $\triangleright \text{safe}((t, v), q(v), q)$, which holds trivially by taking the "value" branch of the safety and showing that $\triangleright(q(v) \sqsubseteq q(v))$ by monotonicity of $\triangleright$ and reflexivity of the view-shift. $\square$

Lemma 27 (Region-Alloc)**.**

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

Proof. By the definition of the view shift, we take any $r \in \mathit{Prop}_{\mathit{TSO}}$, $m \in \mathcal{M}$, $h \in \mathit{Heap}$, $U \in \mathit{SPool}$ such that $m \in P * r$ and $(h, U) \in \lfloor m \rfloor_F$. Hence, there have to exist $l_1, l_2, l_c \in \mathit{LState}$, $s \in \mathit{SState}$, $\zeta \in \mathit{AMod}$ and $\mathit{sr} : \mathit{dom}(s) \cap F \to \mathit{LState}$ such that

$$
\begin{aligned}
&h = l_c.h \qquad l_c = l_1 \bullet_{\mathit{LState}} l_2 \bullet \Pi_{r \in \text{dom}(s) \cap F}\, \mathit{sr}(r) \qquad m = (l_1 \bullet_{\mathit{LState}} l_2, s, U, \zeta)\\
&\forall r \in \text{dom}(s) \cap F.\ (\mathit{sr}(r), U) \in \lfloor (s, \zeta) \rfloor_r \qquad (l_1, s, U, \zeta) \in P \qquad (l_2, s, U, \zeta) \in r
\end{aligned}
$$

Pick an $n \in U$ such that $\forall \alpha \in \mathit{AId}.\ l_c.c(n, \alpha) = 0$ and $n \notin \text{dom}(s) \cup \text{dom}(\zeta)$. Now, let $s' = s[n \mapsto (x, T)]$, $\zeta' = \zeta[n \mapsto \mathit{lam}(\lambda(y, \zeta).\ \{(l, s, U) \in \mathit{AState} \mid (l, s, U, \zeta) \in I(n)(y)\})]$. Note that ζ' is well-defined, by virtue of Lemma 25. We have $s \leq s'$ and $\zeta \leq \zeta'$, and so $(l_1, s', U, \zeta') \in P$ and $(l_2, s', U, \zeta') \in r$. Furthermore, $((\varepsilon, [(n, A) \mapsto 1]), s', U, \zeta') \in \circledast_{\alpha \in A}[\alpha]_1^n$, and so, by the validity assumption,

$$(l_1 \bullet_{\mathit{LState}} (\varepsilon, [(n, A) \mapsto 1]), s', U, \zeta') \in \triangleright I(n)(x).$$

Now, clearly $(\varepsilon, s', U, \zeta') \in \mathit{region}(X, T, n)$. Furthermore, for any $\zeta'' \geq \zeta'$ and $y \in$

SId,

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

Thus, $(\varepsilon, s', U, \zeta'') \in rintr(I,n)$ also holds. Hence, we have

$$(l_2 \bullet_{LState} (\varepsilon, [(n,B) \mapsto 1]), s', U, \zeta') \in region(X,T,n) * rintr(I,n) * (\circledast_{\alpha \in B} [\alpha]^n_1) * r.$$

Lastly, we also get $(h,U) \in \lfloor (l_2 \bullet_{LState} (\varepsilon, [(n,B) \mapsto 1]), s', U, \zeta') \rfloor_F$, which ends the proof. □

Lemma 28.

$$\begin{aligned} &\forall A \in \mathcal{P}(\Delta(RId)), X, Y \in \mathcal{P}(\Delta(SId)), T \in \Delta(LTS), n \in \Delta(RId). \\ &\forall I \in \Delta(SId) \to Prop_{TSO}, p, q \in Prop_{TSO}, \alpha \in \Delta(AId), \pi \in \Delta(Perm^+), f : X \to Y. \\ &\quad \text{stable}(p) \wedge n \in A \wedge (\forall x \in X.\ (x, f(x)) \in T(\alpha) \vee f(x) = x) \wedge \\ &\quad (\forall x \in X.\ p * (\triangleright I(x)) * [\alpha]^n_\pi \sqsubseteq_{A \setminus \{n\}} q * \triangleright I(f(x))) \\ &\qquad \Rightarrow (region(X,T,n) * rintr(I,n) * p * [\alpha]^n_\pi \sqsubseteq_A region(Y,T,n) * q) \end{aligned}$$

Proof. Take $r \in Prop_{TSO}$, $m \in \mathcal{M}$, $h \in Heap$ and $U \in SPool$ such that

$$\underset{A}{\text{stable}}(r) \qquad m \in region(X,T,n) * rintr(I,n) * p * [\alpha]^n_\pi * r \qquad (h,U) \in \lfloor m \rfloor_A.$$

This means, there exist $l_1, l_2, l_3, l_4, l_5 \in LState$, $s \in SState$, $U' \in SPool$ and $\zeta \in AMod$ such that

$$\begin{gathered} (l_1, s, U', \zeta) \in region(X,T,n) \qquad (l_2, s, U', \zeta) \in rintr(I,n) \\ (l_3, s, U', \zeta) \in p \qquad (l_4, s, U', \zeta) \in [\alpha]^n_\pi \qquad (l_5, s, U', \zeta) \in r \\ m = (l_1 \bullet_{LState} l_2 \bullet_{LState} l_3 \bullet_{LState} l_4 \bullet_{LState} l_5, s, U', \zeta). \end{gathered}$$

Let $l = m.l$. By unfolding $(h,U) \in \lfloor m \rfloor_A$ we get that $U = U'$, and $l_c \in LState$ and $sr : \mathrm{dom}(s) \cap A \to LState$ such that

$$h = l_c.h \qquad l_c = l \bullet_{LState} \Pi_{r \in \mathrm{dom}(s) \cap A} sr(r) \qquad \forall r \in \mathrm{dom}(s) \cap A.\ (sr(r), U) \in \lfloor (s,\zeta) \rfloor_r.$$

Since $(l_1, s, U, \zeta) \in region(X,T,n)$, we know $n \in \mathrm{dom}(s)$, $s(n).s \in X$ and $s(n).p = T$ hold. Since $n \in \mathrm{dom}(s) \cap A$, we get $sr(n) \in \lfloor (s,\zeta) \rfloor_n$ and $\pi_2(sr(n)) = U$, and so $(sr(n), s, U) \in app(\zeta(n))(s(n).s, \zeta)$. Unfolding $(l_2, s, U, \zeta) \in rintr(I,n)$, we get that

$$\forall x \in \Delta(SId), \zeta' \geq \zeta.\ app(\zeta(n))(x, \zeta') = \triangleright(\lambda(l,s,U).\ I(x)(l,s,U,\zeta')).$$

From this, instantiating with $s(n).s$ and ζ, we get that $\triangleright(I(s(n).s)(sr(n), s, U, \zeta))$. By assumption, we know that $(s(n).s, f(s(n).s)) \in T(\alpha)$ (in the case where $f(s(n).s) =$

$s(n).s$ it suffices to use the assumed view-shift). Furthermore, since $(l_4, s, \zeta) \in [\alpha]^n_\pi$ it follows that $\pi \leq l_4.c(n, \alpha)$ and thus $l_3.c(n, \alpha) < 1$, $sr(n).c(n, \alpha) < 1$ and $l_5.c(n, \alpha) < 1$. Thus,

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

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

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

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

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

and so, by stability of region interpretations,

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

Thus, we get that

$$\begin{aligned}&(l \bullet_{LState} \pi_1(sr(n)), s', U, \zeta) \in \\ &\qquad \mathit{region}(\{f(s(n).s)\}, T, n) * (\triangleright I(s(n).s)) * [\alpha]^n_\pi * p * r * \{m \in \mathcal{M} \mid \zeta \leq m.a\}\end{aligned}$$

and

$$(h, U) \in \lfloor (l \bullet_{LState} sr(n), s', U, \zeta) \rfloor_{A \setminus \{n\}}.$$

Now we can use the assumed view-shift, from which it follows that there exists an $m' \in \mathcal{M}$ such that

$$m' \in q * (\triangleright I(f(s(n).s))) * r * \mathit{region}(\{f(s(n).s)\}, T, n) * \{m \in \mathcal{M} \mid \zeta \leq m.a\}$$
$$(h, U) \in \lfloor m' \rfloor_{A \setminus \{n\}}.$$

Thus, there exist $l'_1, l'_2, l'_3, l'_4, l'_5 \in \mathit{LState}$, $s'' \in \mathit{SState}$ and $\zeta' \in \mathit{AMod}$ such that

$$m' = (l'_1 \bullet_{LState} l'_2 \bullet_{LState} l'_3 \bullet_{LState} l'_4 \bullet_{LState} l'_5, s'', U, \zeta')$$
$$(l'_1, s'', U, \zeta') \in q \qquad (l'_2, s'', U, \zeta') \in \triangleright I(f(s(n).s)) \qquad (l'_3, s'', U, \zeta') \in r$$
$$(l'_4, s'', U, \zeta') \in \mathit{region}(\{f(s(n).s)\}, T, n) \qquad (l'_5, s'', U, \zeta') \in \{m \in \mathcal{M} \mid \zeta \leq m.a\}.$$

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

$$l'_c = l'_1 \bullet_{LState} l'_2 \bullet_{LState} l'_3 \bullet_{LState} l'_4 \bullet_{LState} l'_5 \bullet_{LState} \circledast_{r \in \mathrm{dom}(s'') \cap (A \setminus \{n\})} \pi_1(sr'(r))$$
$$h = l'_c.h \qquad \forall r \in \mathrm{dom}(s'') \cap (A \setminus \{n\}).\ (sr'(r), U) \in \lfloor (s'', \zeta') \rfloor_r.$$

From $(l'_5, s'', U, \zeta') \in \{m \in \mathcal{M} \mid \zeta \leq m.a\}$ it follows that $\zeta \leq \zeta'$, and thus

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

Hence, $(l'_2, s'', U) \in \mathit{app}(\zeta'(n))(f(s(n).s), \zeta')$, and so $(l'_2, U) \in \lfloor (s'', \zeta') \rfloor_n$. Thus, we have $(h, U) \in \lfloor \mathit{region}(Y, T, n) * q \rfloor_A$, which ends the proof. $\square$

Lemma 29.

$\forall A \in \mathcal{P}(\Delta(RId)).\ \forall X \in \mathcal{P}(\Delta(SId)).\ \forall T \in \Delta(LTS).\ \forall n \in \Delta(RId).$
$\forall I \in \Delta(SId) \rightarrow Prop_{TSO}.\ \forall p, p' \in Prop_{TSO}.\ \forall q \in \Delta(SId) \rightarrow Prop_{TSO}.\ \forall B \in \mathcal{P}(\Delta(AId)).$
$\forall f : \Delta(SId) \rightarrow \mathcal{P}(\Delta(SId)).\ \forall g : \Delta(AId) \rightarrow \Delta(Perm^{+}).$

$$(\forall y \in \Delta(SId).\ \text{stable}(q(y))) \wedge n \in A \wedge (\forall x \in X.\ \forall y \in f(x).\ (x, y) \in T(B)^{*}) \wedge$$
$$(\forall x \in X.\ a\ \text{sat}_{A \setminus \{n\}}\ \{p * (\triangleright I(x))\} \left\{\exists y \in f(x).\ q(y) * \circledast_{\alpha \in B} [\alpha]^{n}_{g(\alpha)} * \triangleright I(y)\right\})$$
$$\Rightarrow a\ \text{sat}_{A}\ \{region(X, T, I, n) * p\} \left\{\exists y \in \Delta(SId).\ region(\{y\}, T, n) * \circledast_{\alpha \in B} [\alpha]^{n}_{g(\alpha)} * q(y)\right\}$$

Proof. Assume $r \in Prop_{TSO}$, $m \in \mathcal{M}$, $h, h' \in \Delta(Heap)$ and $U, U' \in SPool$ such that

$$m \in region(X, T, I, n) * p * \triangleright r \quad (h, U) \in \lfloor m \rfloor_{A} \quad (h', U') \in [\![a]\!](h, U) \quad \underset{A}{\text{stable}}(r).$$

Thus, there exist $l_1, l_2, l_3, l_4 \in LState$, $s \in SState$ and $\zeta \in AMod$ such that

$$(l_1, s, U, \zeta) \in region(X, T, n) \quad (l_2, s, U, \zeta) \in rintr(I, n) \quad (l_3, s, U, \zeta) \in p$$
$$(l_4, s, U, \zeta) \in \triangleright r \quad m = (l_1 \bullet l_2 \bullet l_3 \bullet l_4 \bullet l_5, s, U, \zeta).$$

Let $l = l_1 \bullet l_2 \bullet l_3 \bullet l_4$. Unfolding $(h, U) \in \lfloor m \rfloor_{A}$ we get an $l_c \in LState$ and $sr : (\text{dom}(s) \cap A) \rightarrow LState$ such that

$$h = l_c.h \quad l_c = l \bullet \circledast_{r \in (\text{dom}(s) \cap A)} sr(r) \quad \forall r \in (\text{dom}(s) \cap A).\ (sr(r), U) \in \lfloor (s, \zeta) \rfloor_{r}.$$

Since $(l_1, s, U, \zeta) \in region(X, T, n)$, it follows that $n \in \text{dom}(s)$, $s(n).s \in X$ and $s(n).p = T$. Since we also know that $n \in A$, it follows that $(sr(n), U) \in \lfloor (s, \zeta) \rfloor_{n}$, and so $(sr(n), s, U) \in app(\zeta(n))(s(n).s, \zeta)$. Unfolding $(l_2, s, U, \zeta) \in rintr(I, n)$ it follows that

$$\forall x \in \Delta(SId).\ \forall \zeta' \geq \zeta.\ app(\zeta(n))(x, \zeta') = \triangleright(\lambda(l, s, U).\ I(x)(l, s, U, \zeta')).$$

Thus, in particular, $app(\zeta(n))(s(n).s, \zeta) = \triangleright(\lambda(l, s, U).\ I(s(n).s)(l, s, U, \zeta))$, and so

$$\triangleright I(s(n).s)(sr(n), s, U, \zeta).$$

The high-level plan is now to use the assumption with a slightly modified state to account for opening of the region, and afterwards use the state obtained from the assumption to build a final instrumented state. To this end, we instantiate the assumption with a frame $r' = r * region(\{s(n).s\}, T, n) * \{m \in \mathcal{M} \mid m.\zeta \geq \zeta\}$ and an instrumented state $\bar{m} = (l \bullet sr(n), s, U, \zeta)$. The erasure $(h, U) \in \lfloor \bar{m} \rfloor_{A \setminus \{n\}}$ holds easily, since $n \notin A \setminus \{n\}$ but $sr(n)$ is in local state. We also know that $(l_3, s, U, \zeta) \in p$, $(sr(n), s, U, \zeta) \in \triangleright(I(s(n).s))$ and $(l_4, s, U, \zeta) in \triangleright r$, so it suffices to show that

$$(l_1 \bullet l_2, s, U, \zeta) \in \triangleright(region(\{s(n).s\}, T, n) * \{m \in \mathcal{M} \mid m.\zeta \geq \zeta\}),$$

which holds by monotonicity of later and the definition. The frame is also stable, since we cannot change the state of region n.

This means we get a state $\bar{m}'$ such that $(h', U') \in \triangleright\lfloor \bar{m}' \rfloor_{A \setminus \{n\}}$ and

$$\bar{m}' \in \triangleright(\exists y \in \mathsf{f}(x) * q(y) * \circledast_{\alpha \in B}[\alpha]^n_{g(\alpha)} * \triangleright I(y) * r').$$

Thus, we can take a $y \in \Delta(SId)$ such that $\triangleright(y \in f(x))$, and local states l'_1, l'_2, l'_3 and l'_4, such that

$$\bar{m}' = (l'_1 \bullet l'_2 \bullet l'_3 \bullet l'_4, s', U', \zeta')$$
$$(l'_1, s', U', \zeta') \in \triangleright q(y) \quad (l'_2, s', U', \zeta') \in \triangleright \circledast_{\alpha \in B} [\alpha]^n_{g(\alpha)} \quad \triangleright ((l'_3, s', U', \zeta') \in \triangleright I(y))$$
$$(l'_4, s', U', \zeta') \in \triangleright r \quad \triangleright (s'(n).s = s(n).s) \quad \triangleright (s'(n).p = T) \quad \triangleright (\zeta' \geq \zeta).$$

Now, we can take $m' = (l'_1 \bullet l'_2 \bullet l'_4, s'[n \mapsto (y, T)], U', \zeta')$ as the witness. We need to show that $\triangleright((h', U') \in \lfloor m' \rfloor_A)$ and $m' \in \triangleright(\exists y \in \Delta(SId).\, region(\{y\}, T, n) * q(y) * \circledast_{\alpha \in B}[\alpha]^n_{g(\alpha)} * r)$. For the latter, we take y as the witness and split the local state four ways into ε, l'_1, l'_2 and l'_4. The region assertion holds by definition, and the assertion about permissions by the earlier property, since it doesn't depend on the shared state. That leaves us with r and $q(y)$. Both these assertions are stable, so it suffices to show that x and y are in the interference relation from their perspective. To show this, notice that the permissions in B suffice to transition from x to y – and these permissions are outside both l'_1 and l'_4. Thus, we can get $(l'_1, s'[n \mapsto (y, T)], U', \zeta') \in \triangleright q(y)$ and $(l'_4, s'[n \mapsto (y, T)], U', \zeta') \in \triangleright r$.

This leaves us with proving that $(h', U') \in \lfloor m' \rfloor_A$. To this end, take $\bar{l}$ and $\bar{sr} : \mathrm{dom}(s') \cap (A \setminus \{n\}) \to LState$ – the witnesses of $(h', U') \in \lfloor \bar{m}' \rfloor_{A \setminus \{n\}}$. We know that

$$h' = \bar{l}.h \quad \bar{l} = l'_1 \bullet l'_2 \bullet l'_3 \bullet l'_4 \bullet \Pi_{r \in \mathrm{dom}(s) \cap (A \setminus \{n\})} \bar{sr}(r)$$
$$\forall r \in \mathrm{dom}(s) \cap (A \setminus \{n\}).\ \triangleright ((\bar{sr}(r), U') \in \lfloor (s', \zeta') \rfloor_r)$$

Take as witnesses $l' = \bar{l}$ and $sr' = \bar{sr}[n \mapsto l'_3]$. Clearly sr' has the appropriate domain, and the first two conditions hold. For the final one, take any $r \in \mathrm{dom}(s) \cap A$. We have two cases to consider: either $r = n$, or $r \in \mathrm{dom}(s) \cap (A \setminus \{n\})$. In the latter, we can use the same argument as for r and $q(y)$ to show that the update of shared state is an allowed interference for the region r. In the former, we need to show that $\triangleright(l'_3, U') \in \lfloor (s'[n \mapsto (y, T)], \zeta') \rfloor_r$. Unrolling the definition, it suffices to show that $\triangleright(l'_3, s'[n \mapsto (y, T)], U') \in app(\zeta'(r))(y, \zeta')$. However, since $\triangleright(\zeta' \geq \zeta)$, by an earlier equation that followed from region interpretation, it suffices to show that $\triangleright(l'_3, s'[n \mapsto (y, T)], U', \zeta') \in \triangleright I(y)$, which follows by the same stability argument as the other assertions. □

Lemma 30. *For any $p \in Prop_{SC}$, $(l, s, U, \zeta) \in \ulcorner p \urcorner$, $(l', U') \in R_{sb}(l, U)$, $(l', s, U', \zeta) \in \ulcorner p \urcorner$.*

Proof. To show the lemma holds, it suffices to show p is closed under each of R_n, R_w and R_f. These proofs are presented in the following paragraphs.

(R_n) Take any $(l, s, U, \zeta) \in \ulcorner p \urcorner$ and $(l', U') \in R_n(l, U)$. By definition of R_n we have that $l = l'$, $\forall t \in \mathrm{dom}(U') \setminus \mathrm{dom}(U).\ U'(t) = \varepsilon$ and $\forall t \in \mathrm{dom}(U).\ U'(t) = U(t)$. From definition of $\ulcorner - \urcorner$ we get a state $l_1 \leq l$ such that $(l_1, s, \zeta) \in p$ and

$$\forall t \in \mathrm{dom}(U).\ \forall (o, f) \in \mathrm{dom}(l_1).\ \forall v \in \mathit{Val}.\ (o, f, v) \notin U(t).$$

We take the same l_1 as a witness, and so we only need to show that $(o, f, v) \notin U'(t)$ for any $t \in \mathrm{dom}(U')$, $(o, f) \in \mathrm{dom}(l_1)$ and $v \in \mathit{Val}$. If $t \in \mathrm{dom}(U)$, the property holds, since in that case $U(t) = U(t')$ and we know it held for U. If $t \notin \mathrm{dom}(U)$, on the other hand, we know $U'(t) = \varepsilon$, so in particular $(o, f, v) \notin U'(t)$.

(R_w) Take any $(l, s, U, \zeta) \in \ulcorner p \urcorner$ and $(l', U') \in R_w(l, U)$. By definition of R_w we know that $l = l'$, $\mathrm{dom}(U) = \mathrm{dom}(U')$ and we get $t \in \mathrm{dom}(U)$, $o \in \mathit{OId}$, $f \in \mathit{FName}$, and $v \in \mathit{Val}$ such that $(o, f) \notin \mathrm{dom}(l)$, $U'(t) = U(t) \cdot (o, f, v)$ and $U'(t') = U(t')$ for any $t' \neq t$. From the definition of $\ulcorner - \urcorner$ we get a local state $l_1 \leq l$ such that $(l_1, s, \zeta) \in p$ and

$$\forall t \in \mathrm{dom}(U).\ \forall (o, f) \in \mathrm{dom}(l_1).\ \forall v \in \mathit{Val}.\ (o, f, v) \notin U(t).$$

Again, we take l_1 as the witness, which means we only need to prove $(o', f', v') \notin U(t')$ for any $(o', f') \in \mathrm{dom}(l_1)$, $t' \in \mathrm{dom}(U)$. Clearly, it suffices to check that this holds for the newly added update, (o, f, v) in thread t, since the rest of the contents of U' is inherited from U. However, we know that $(o, f) \notin \mathrm{dom}(l)$, so clearly $(o, f) \notin \mathrm{dom}(l_1)$, so the property holds.

(R_f) Take any $(l, s, U, \zeta) \in \ulcorner p \urcorner$ and $(l', U') \in R_w(l, U)$. By definition of R_w we know that $\mathrm{dom}(U) = \mathrm{dom}(U')$ and we get $t \in \mathrm{dom}(U)$, $o \in \mathit{OId}$, $f \in \mathit{FName}$ and $v \in \mathit{Val}$ such that $l' = \mathrm{flush}(l, (o, f, v))$, $U(t) = (o, f, v) \cdot U'(t)$ and $U(t') = U'(t')$ for any $t' \neq t$. As before, from the definition of $\ulcorner - \urcorner$ we get an $l_1 \leq l$ such that $(l_1, s, \zeta) \in p$ and there are no updates to the domain of l_1. From the latter property, we know that $(o, f) \notin \mathrm{dom}(l_1)$, and so $l_1 \leq l'$. This allows us to take l_1 as the witness, and in this case both properties hold trivially, since the updates in U' are a subset of those in U. □

2.11 Thread-pool Evaluation

$$\begin{aligned} \mathit{eval} &: \Delta(\mathit{MSt}) \times \Delta(\mathit{TPool}) \times \mathcal{P}(\Delta(\mathit{MSt}) \times \Delta(\mathit{TPool})) \to \Omega \\ \mathit{eval}(\mu, T, q) &\stackrel{\text{def}}{=} (\mathrm{irr}(\mu, T) \Rightarrow (\mu, T) \in q)\ \wedge (\forall T', \mu'.\ (\mu, T) \to (\mu', T') \Rightarrow \triangleright \mathit{eval}(\mu', T', q)) \end{aligned}$$

Lemma 31 (Safe-Eval). *For any program state $(\mu, T) \in \Delta(\mathit{PSt})$ and families of assertions $p : \mathrm{dom}(T) \to \mathit{Prop}_{\mathit{TSO}}$, $q : \mathrm{dom}(T) \to \mathit{Val} \to \mathit{Prop}_{\mathit{TSO}}$, if*

$$\forall t \in \mathrm{dom}(T).\ \mathrm{stable}(p(t)) \qquad \forall t \in \mathrm{dom}(T).\ \forall v \in \mathit{Val}.\ \mathrm{stable}(q(t)(v)))$$
$$\forall t \in \mathrm{dom}(T).\ \mathrm{safe}((t, T(t)), p(t), q(t)) \qquad \mu \in \lfloor \circledast_{t \in \mathrm{dom}(T)} p(t) \rfloor$$

then

$$\mathit{eval}\left(\mu, T, \lambda(\mu', T').\ \mu' \in \lfloor \circledast_{t \in \mathrm{dom}(T)} q(t)(T'(t)) \rfloor\right).$$

Proof. By Löb induction. Since μ is an erasure of a state, we know there exist h and U such that $\mu = (h, U)$ and $\mathrm{dom}(U) = \mathrm{dom}(T)$. By Lemma 1, there are two cases: either we are in a terminal state, or there exist a possible reduction. We proceed by inspecting these two.

Let us first consider the terminal case. By Lemma 1, we know that for any thread $t \in \mathrm{dom}(T)$, $T(t)$ is a value, and $U(t) = \varepsilon$. Thus, the configuration is irreducible, so the right conjunct holds trivially. In the left conjunct, we exhibit $\mathrm{irr}(\mu, T)$ and are left with showing that $(h, U) \in \lfloor \circledast_{t \in \mathrm{dom}(T)} q(t)(T(t)) \rfloor$. However, since we know that no thread can proceed, we also learn, from the safety assumption, that for any thread we have $p(t) \sqsubseteq q(t)(T(t))$. This shows us how to proceed: we should successively apply the view-shifts for each consecutive thread, while treating the separating conjunctions of the threads already shifted and these yet to be shifted as a frame. Formally we need to weaken our assumption to say that for any thread pools T_1 and T_2 such that $T_1 \uplus T_2 = T$ if $(h, U) \in \lfloor \circledast_{t \in \mathrm{dom}(T_1)} q(t)(T_1(t)) * \circledast_{t \in \mathrm{dom}(T_2)} p(t) \rfloor$, then $(h, U) \in \lfloor \circledast_{t \in \mathrm{dom}(T)} q(t)(T(t)) \rfloor$, and proceed by induction on the size of T_2. In the base case T_2 is empty, so $T_1 = T$, and the lemma holds trivially. Otherwise, we know that $T_2 = (t, v) \uplus T_2'$ for some T_2', t and v. Thus, since all of p and q are stable, we can apply the view-shift for thread t: we know that $(h, U) \in \lfloor p(t) * \circledast_{t \in \mathrm{dom}(T_1)} q(t)(T_1(t)) * \circledast_{t \in \mathrm{dom}(T_2')} p(t) \rfloor$, so we get that $(h, U) \in \lfloor q(t)(v) * \circledast_{t \in \mathrm{dom}(T_1)} q(t)(T_1(t)) * \circledast_{t \in \mathrm{dom}(T_2')} p(t) \rfloor$. Now, to use the induction hypothesis for the smaller T_2' we only need to take $T_1' = T_1 \uplus (t, v)$. Thus, this part of the proof is finished.

Let us now consider the case where a reduction could occur. The decomposition lemma gives us a state (μ', T') such that $((h, U), T) \to (\mu', T')$. Thus, the left conjunct holds trivially. On the right, by definition of single step evaluation, we have a thread $t \in \mathrm{dom}(T)$, an action $a \in \mathit{Act}$, and a thread pool $T' \in \mathit{TPool}$, such that $(t, T(t)) \xrightarrow{a} T''$, $\mu' \in [\![a]\!](\mu)$ and $T' = (T - t) \uplus T''$. We will consider the case when $a = \mathit{flush}(t)$ separately, since this can occur even if $T(t)$ is a value; all the other actions will follow from the safety of thread t.

Let us assume $a = \mathit{flush}(t)$. In this case we can take the instrumented state m that mediates the erasure and use the Lemma 19 to get a state $m' \in \triangleright \circledast_{t \in \mathrm{dom}(T)} p(t)$ such that $\mu' \in \triangleright \lfloor m' \rfloor$. The remaining obligation is easily discharged by induction hypothesis and monotonicity of $\triangleright$.

Now, let us consider the case when a is not a flush action. In this case, we know that $T(t)$ can not be a value, so from the assumption of safety of t we get $p' : \mathrm{dom}(T'') \to \mathit{Prop}_{\mathit{TSO}}$ and the following facts:

$$t \in \mathrm{dom}(T'') \quad \forall t' \in \mathrm{dom}(T'').\ \triangleright \mathrm{stable}(p'(t')) \quad a \text{ sat } \{p(t)\} \left\{ \circledast_{t' \in \mathrm{dom}(T'')} p'(t') \right\}$$
$$\triangleright \mathrm{safe}((t, T''(t)), p'(t), q(t)) \quad \forall t' \in \mathrm{dom}(T'' - t).\ \mathrm{safe}((t', T''(t')), p'(t'), \lambda_.\top).$$

As in the previous case, we can now take the mediating instrumented state m and use the atomic satisfaction, taking $\circledast_{t' \in \mathrm{dom}(T-t)} p(t')$ as a frame, to obtain a state m' such that

$$\mu' \in \triangleright \lfloor m' \rfloor \qquad m' \in \triangleright \circledast_{t' \in \mathrm{dom}(T-t)} p(t) * \circledast_{t' \in \mathrm{dom}(T'')} p'(t)$$

This, along with the previously obtained safety and stability properties allows us to use the induction hypothesis to conclude that

$$\triangleright eval(\mu', T', \lambda(\mu'', T'').\ \mu'' \in \lfloor \circledast_{t \in \mathrm{dom}(T)} q(t)(T''(t)) * \circledast_{t \in \mathrm{dom}(T''-t)} \top \rfloor).$$

However, since we know that $p * \top \subseteq \top$ for any assertion p, this gives us the required obligation that

$$\triangleright eval(\mu', T', \lambda(\mu'', T'').\ \mu'' \in \lfloor \circledast_{t \in \mathrm{dom}(T)} q(t)(T''(t)) \rfloor),$$

which ends the proof. □

Additional logical connectives

$$\begin{aligned}
\ulcorner p\ \mathbf{in}\ t \urcorner &\stackrel{\mathrm{def}}{=} \{(l, s, U, \zeta) \in \mathcal{M} \mid (\mathrm{flush}(l, U(t)), s, U[t \mapsto \varepsilon], \zeta) \in \ulcorner p \urcorner\} \\
p\ \mathcal{U}_t\ q &\stackrel{\mathrm{def}}{=} \{(l, s, U, \zeta) \in \mathcal{M} \mid \exists \alpha, \beta \in \Delta(\mathit{SBuffer}).\ \exists o \in \Delta(\mathit{OId}).\ \exists f \in \Delta(\mathit{FName}).\ \exists v \in \Delta(\mathit{Val}). \\
&\qquad U(t) = \alpha \cdot (o, f, v) \cdot \beta \wedge (l, s, U[t \mapsto \alpha], \zeta) \in p \wedge \\
&\qquad (\mathrm{flush}(l, \alpha \cdot (o, f, v)), s, U[t \mapsto \beta], \zeta) \in q\} \\
\underset{A}{\mathrm{cl}}(p) &\stackrel{\mathrm{def}}{=} \{m \in \mathcal{M} \mid \exists m' \in p.\ m'\ (\leq \cup R_A)^*\ m\}
\end{aligned}$$

Semantic read and flush judgments

$$\begin{aligned}
p \vdash_{\mathrm{fl}(t)} q &\stackrel{\mathrm{def}}{=} \forall (l, s, U, \zeta) \in p.\ t \in \mathrm{dom}(U) \Rightarrow \triangleright(\mathrm{flush}(l, U(t)), s, U[t \mapsto \varepsilon], \zeta) \in q \\
p \vdash_{\mathrm{rd}(t)} o.f \mapsto v &\stackrel{\mathrm{def}}{=} \forall (l, s, U, \zeta) \in p.\ t \in \mathrm{dom}(U) \Rightarrow \triangleright \mathrm{lookup}(o.f, U(t), l) = v
\end{aligned}$$

3 Logic

The specification logic is given by the specification entailment judgment,

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

where S is a specification Γ — a logical variable context, and Φ — a specification context. The specification logic extends a standard higher-order intuitionistic logic.Similarly, both of the assertion logics, SC and TSO, are given by the assertion logic entailment judgments, of the form

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

where L ranges over $\{SC, TSO\}$, P and Q are the assertions of the appropriate logic, Γ is the logical variable context, Δ — the program variable context, and Φ — the specification context. The SC logic is a standard higher order assertion logic, while the TSO logic extends the higher-order intuitionistic separation logic with several additional proof rules for the TSO connectives.

3.1 TSO Assertion Logic

Listed below are some of the proof rules of the TSO assertion logic. For the sake of clarity, we do not present the introduction and elimination rules for the standard connectives of higher-order intuitionistic separation logic here.

$$\frac{\Gamma; \Delta \mid \Phi \vdash \mathsf{stable}(\mathsf{P}) \wedge \mathsf{stable}(\mathsf{Q})}{\Gamma; \Delta \mid \Phi \mid (\!|\mathsf{P}\,\mathcal{U}_t\,\mathsf{Q}|\!) \vdash_{TSO} (\mathsf{P}\,\mathcal{U}_t\,\mathsf{Q}) \vee \mathsf{Q}}\ \textsc{Stab-}\mathcal{U}$$

$$\frac{}{\Gamma; \Delta \mid \Phi \mid \mathsf{P} \vdash_{TSO} (\!|\mathsf{P}|\!)}\ \textsc{Stab-I} \qquad \frac{\Gamma; \Delta \mid \Phi \vdash \mathsf{stable}(\mathsf{P})}{\Gamma; \Delta \mid \Phi \mid (\!|\mathsf{P}|\!) \vdash_{TSO} \mathsf{P}}\ \textsc{Stab-S}$$

$$\frac{}{\Gamma; \Delta \mid \Phi \mid \ulcorner\mathsf{P}\urcorner \vdash_{TSO} \ulcorner\mathsf{P}\ \mathbf{in}\ \mathsf{t}\urcorner}\ \text{W-H}\textsc{Interp} \qquad \frac{}{\Gamma; \Delta \mid \mathsf{P}\,\mathcal{U}_t\,\mathsf{Q} \vdash_{TSO} \ulcorner\mathsf{Q}\ \mathbf{in}\ \mathsf{t}\urcorner}\ \text{W-}\mathcal{U}$$

$$\frac{\Gamma; \Delta \mid \Phi \mid \mathsf{P}_1 \vdash_{TSO} \mathsf{P}_2 \qquad \Gamma; \Delta \mid \Phi \mid \mathsf{Q}_1 \vdash_{TSO} \mathsf{Q}_2}{\Gamma; \Delta \mid \Phi \mid \mathsf{P}_1\,\mathcal{U}_t\,\mathsf{Q}_1 \vdash_{TSO} \mathsf{P}_2\,\mathcal{U}_t\,\mathsf{Q}_2}\ \textsc{Cons-}\mathcal{U}$$

$$\frac{\Gamma; \Delta \mid \Phi \vdash \mathsf{b\text{-}stable}(\mathsf{P})}{\Gamma; \Delta \mid \Phi \mid \mathsf{P} * (\mathsf{Q}\,\mathcal{U}_t\,\mathsf{R}) \vdash_{TSO} \mathsf{Q}\,\mathcal{U}_t\,(\mathsf{P} * \mathsf{R})}\ *\text{-}\mathcal{U}$$

3.2 Syntactic Sugar

We provide some syntactic sugar for more convenient reasoning within the TSO logic, where the pre- and postconditions are parametrized with the current thread identifier.

$$\begin{aligned}
\overline{\mathsf{P}} &\equiv \lambda \mathsf{t} \in \mathsf{TId}.\ \ulcorner\mathsf{P}\ \mathbf{in}\ \mathsf{t}\urcorner \\
\mathsf{P}\,\mathcal{U}\,\mathsf{Q} &\equiv \lambda \mathsf{t} \in \mathsf{TId}.\ \mathsf{P}(\mathsf{t})\,\mathcal{U}_{\mathsf{t}}\,\mathsf{Q}(\mathsf{t}) \\
\mathsf{P}\,\mathcal{U}^{\mathrm{o}}\,\mathsf{Q} &\equiv \lambda \mathsf{t} \in \mathsf{TId}.\ \exists \mathsf{t}' \in \mathsf{TId}.\ \mathsf{t} \neq \mathsf{t}' * \mathsf{P}(\mathsf{t}')\,\mathcal{U}_{\mathsf{t}'}\,\mathsf{Q}(\mathsf{t}') \\
\mathsf{stable}(\mathsf{P}) &\equiv \mathsf{r\text{-}stable}(\mathsf{P}) \wedge \mathsf{b\text{-}stable}(\mathsf{P}) \\
\mathsf{iam}(\mathsf{t}') &\equiv \lambda \mathsf{t} \in \mathsf{TId}.\ \mathsf{t} = \mathsf{t}'
\end{aligned}$$

3.3 Read and Flush Rules

In the language, we have multiple constructs that either read values from the store, or affect the state of the store buffers. We also have extra connectives, which do not admit simple, small-footprint specifications (i.e., until). Hence, we provide two additional judgments, that can be used to determine these effects. The read judgment determines what values can be observed by thread t, while the flush judgment describes the effect of flushing the store buffer of thread t on an assertion. Both of these are explicitly parameterized with thread t, such that they can be used together — as they will in one of the CAS rules.

The forms of these judgments are $\Gamma; \Delta \mid \mathsf{P} \vdash_{\mathrm{rd}(t)} \mathsf{x.f} \mapsto \mathsf{v}$ and $\Gamma; \Delta \mid \mathsf{P} \vdash_{\mathrm{fl}(t)} \mathsf{Q}$, respectively, with P and Q ranging over $\mathsf{Prop}_{\mathsf{TSO}}$.

$$\frac{}{\Gamma; \Delta \mid \triangleright\ulcorner \mathsf{x.f} \mapsto \mathsf{v} \textbf{ in } t\urcorner \vdash_{\mathrm{rd}(t)} \mathsf{x.f} \mapsto \mathsf{v}}\ \textsc{Rd-Ax} \qquad \frac{\Gamma; \Delta \mid \cdot \mid \mathsf{P} \vdash \mathsf{Q} \qquad \Gamma; \Delta \mid \mathsf{Q} \vdash_{\mathrm{rd}(t)} \mathsf{x.f} \mapsto \mathsf{v}}{\Gamma; \Delta \mid \mathsf{P} \vdash_{\mathrm{rd}(t)} \mathsf{x.f} \mapsto \mathsf{v}}\ \textsc{Rd-Cons}$$

$$\frac{\Gamma; \Delta \mid \mathsf{Q} \vdash_{\mathrm{rd}(t)} \mathsf{x.f} \mapsto \mathsf{v}}{\Gamma; \Delta \mid \mathsf{P}\ \mathcal{U}_t\ \mathsf{Q} \vdash_{\mathrm{rd}(t)} \mathsf{x.f} \mapsto \mathsf{v}}\ \textsc{Rd-}\mathcal{U}\textsc{-Eq} \qquad \frac{\Gamma; \Delta \mid \mathsf{P} \vdash_{\mathrm{rd}(t)} \mathsf{x.f} \mapsto \mathsf{v} \qquad t \neq t'}{\Gamma; \Delta \mid \mathsf{P}\ \mathcal{U}_{t'}\ \mathsf{Q} \vdash_{\mathrm{rd}(t)} \mathsf{x.f} \mapsto \mathsf{v}}\ \textsc{Rd-}\mathcal{U}\textsc{-NEq}$$

$$\frac{}{\Gamma; \Delta \mid \triangleright\ulcorner \mathsf{P} \textbf{ in } t\urcorner \vdash_{\mathrm{fl}(t)} \ulcorner P\urcorner}\ \textsc{Fl-Ax} \qquad \frac{\Gamma; \Delta \mid \mathsf{P} \vdash_{\mathrm{fl}(t)} \mathsf{P}' \qquad \Gamma; \Delta \mid \mathsf{Q} \vdash_{\mathrm{fl}(t)} \mathsf{Q}'}{\Gamma; \Delta \mid \mathsf{P} * \mathsf{Q} \vdash_{\mathrm{fl}(t)} \mathsf{P}' * \mathsf{Q}'}\ \textsc{Fl-}*$$

$$\frac{\Gamma; \Delta \mid \mathsf{Q} \vdash_{\mathrm{fl}(t)} \mathsf{R}}{\Gamma; \Delta \mid \mathsf{P}\ \mathcal{U}_t\ \mathsf{Q} \vdash_{\mathrm{fl}(t)} \mathsf{R}}\ \textsc{Fl-}\mathcal{U}\textsc{-Eq} \qquad \frac{\Gamma; \Delta \mid \mathsf{P} \vdash_{\mathrm{fl}(t)} \mathsf{P}' \qquad t \neq t'}{\Gamma; \Delta \mid \mathsf{P}\ \mathcal{U}_{t'}\ \ulcorner\mathsf{Q}\urcorner \vdash_{\mathrm{fl}(t)} \mathsf{P}'\ \mathcal{U}_{t'}\ \ulcorner\mathsf{Q}\urcorner}\ \textsc{Fl-}\mathcal{U}\textsc{-NEq}$$

$$\frac{\Gamma; \Delta \mid \cdot \mid \mathsf{P} \vdash \mathsf{P}' \qquad \Gamma; \Delta \mid \mathsf{P}' \vdash_{\mathrm{fl}(t)} \mathsf{Q}' \qquad \Gamma; \Delta \mid \cdot \mid \mathsf{Q}' \vdash \mathsf{Q}}{\Gamma; \Delta \mid \mathsf{P} \vdash_{\mathrm{fl}(t)} \mathsf{Q}}\ \textsc{Fl-Cons}$$

3.4 Specification Logic

In the following we present the rules for reasoning within the specification logic. The introduction and elimination rules for standard higher-order intuitionistic logic are omitted, since they are completely standard.

3.4.1 Reasoning about Hoare triples

Atomic Command Rules

$$\frac{\Gamma; \Delta \mid \mathsf{P}(t) \vdash_{\mathrm{rd}(t)} \mathsf{x.f} \mapsto \mathsf{v}}{\Gamma; \Delta \mid \Phi \vdash \langle \mathsf{P} * \mathsf{iam}(t) \rangle\ \mathsf{x.f}\ \langle \mathsf{r}.\ \mathsf{P} * \mathsf{r} = \mathsf{v} \rangle^{C}} \ \text{A-READ}$$

$$\frac{\Gamma; \Delta \mid \mathsf{P}(t) \vdash_{\mathrm{fl}(t)} \ulcorner \mathsf{x.f} \mapsto - \urcorner * \mathsf{Q}(t) \qquad \mathsf{b\text{-}stable}(\mathsf{Q}(t))}{\Gamma; \Delta \mid \Phi \vdash \langle \mathsf{P} * \mathsf{iam}(t) \rangle\ \mathsf{x.f} := \mathsf{v}\ \langle _.\ \mathsf{P}\ \mathcal{U}\ (\mathsf{Q} * \ulcorner \mathsf{x.f} \mapsto \mathsf{v} \urcorner) \rangle^{C}} \ \text{A-WRITE}$$

$$\frac{\Gamma; \Delta \mid \mathsf{P}(t) \vdash_{\mathrm{rd}(t)} \mathsf{x.f} \mapsto \mathsf{v} \qquad \Gamma; \Delta \mid \mathsf{P}(t) \vdash_{\mathrm{fl}(t)} \mathsf{Q}(t) \qquad \mathsf{v} \neq \mathsf{v_o}}{\Gamma; \Delta \mid \Phi \vdash \langle \mathsf{P} * \mathsf{iam}(t) \rangle\ \mathbf{CAS}(\mathsf{x.f}, \mathsf{v_n}, \mathsf{v_o})\ \langle \mathsf{r}.\ \mathsf{r} = \mathbf{false} * \mathsf{Q} \rangle^{C}} \ \text{A-CAS-FALSE}$$

$$\frac{\Gamma; \Delta \mid \mathsf{P}(t) \vdash_{\mathrm{fl}(t)} \ulcorner \mathsf{x.f} \mapsto \mathsf{v_o} \urcorner * \mathsf{Q}(t) \qquad \mathsf{b\text{-}stable}(\mathsf{Q}(t))}{\Gamma; \Delta \mid \Phi \vdash \langle \mathsf{P} * \mathsf{iam}(t) \rangle\ \mathbf{CAS}(\mathsf{x.f}, \mathsf{v_n}, \mathsf{v_o})\ \langle \mathsf{r}.\ \mathsf{r} = \mathbf{true} * \mathsf{Q} * \ulcorner \mathsf{x.f} \mapsto \mathsf{v_n} \urcorner \rangle^{C}} \ \text{A-CAS-TRUE}$$

$$\frac{\Gamma; \Delta \mid \mathsf{P}(t) \vdash_{\mathrm{fl}(t)} \mathsf{Q}(t)}{\Gamma; \Delta \mid \Phi \vdash \langle \mathsf{P} * \mathsf{iam}(t) \rangle\ \mathbf{fence}\ \langle \mathsf{Q} \rangle} \ \text{A-FENCE}$$

$$\frac{\begin{array}{c} \Gamma, \Delta \mid \Phi \vdash \forall \mathsf{y}, \mathsf{r}.\ \mathsf{stable}(\mathsf{Q}(\mathsf{y}, \mathsf{r})) \\ \Gamma, \Delta \mid \Phi \vdash \mathsf{n} \in C \qquad \Gamma, \Delta \mid \Phi \vdash \forall \mathsf{x} \in X.\ \forall \mathsf{y} \in f(\mathsf{x}).\ (\mathsf{x}, \mathsf{y}) \in (T(A))^{*} \\ \Gamma \mid \Phi \vdash \forall \mathsf{x} \in X.\ (\Delta).\langle \mathsf{P} * \triangleright I(\mathsf{x}) \rangle\ \mathsf{e}\ \langle \mathsf{r}.\ \exists \mathsf{y} \in f(\mathsf{x}).\ \mathsf{Q}(\mathsf{y}, \mathsf{r}) * \circledast_{\alpha \in A} [\alpha]^{\mathsf{n}}_{g(\alpha)} * \triangleright I(\mathsf{y}) \rangle^{C \setminus \{\mathsf{n}\}} \end{array}}{\begin{array}{l} \Gamma \mid \Phi \vdash (\Delta).\ \langle \mathsf{P} * \mathsf{region}(X, T, I, \mathsf{n}) \rangle \\ \qquad \mathsf{e} \\ \qquad \langle \mathsf{r}.\ \exists \mathsf{y}.\ \mathsf{Q}(\mathsf{y}, \mathsf{r}) * \circledast_{\alpha \in A} [\alpha]^{\mathsf{n}}_{g(\alpha)} * \mathsf{region}(\{\mathsf{y}\}, T, I, \mathsf{n}) \rangle^{C} \end{array}} \ \text{A-REGION}$$

$$\frac{\Gamma \mid \Phi \vdash (\Delta).\langle \mathsf{P} \rangle\ \mathsf{e}\ \langle \mathsf{Q} \rangle^{C} \qquad \mathsf{atomic}(\mathsf{e})}{\Gamma \mid \Phi \vdash (\Delta).[\mathsf{P}]\ \mathsf{e}\ [\mathsf{Q}]} \ \text{A-START}$$

$$\frac{}{\mathsf{atomic}(\mathbf{fence})} \qquad \frac{}{\mathsf{atomic}(\mathsf{x.f})} \qquad \frac{}{\mathsf{atomic}(\mathsf{x.f} := \mathsf{v})} \qquad \frac{}{\mathsf{atomic}(\mathbf{CAS}(\mathsf{x.f}, \mathsf{v_n}, \mathsf{v_o}))}$$

Structural Rules

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

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

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

$$\frac{\Gamma \mid \Phi \vdash \mathsf{P}_1 \sqsubseteq^{A} \mathsf{P}_2 \qquad \Gamma \mid \Phi \vdash (\Delta).\langle \mathsf{P}_2 \rangle\ \mathsf{e}\ \langle \mathsf{Q}_2 \rangle^{A} \qquad \Gamma \mid \Phi \vdash \mathsf{Q}_2 \sqsubseteq^{A} \mathsf{Q}_1}{\Gamma \mid \Phi \vdash (\Delta).\langle \mathsf{P}_1 \rangle\ \mathsf{e}\ \langle \mathsf{Q}_1 \rangle^{A}}\ \textsc{A-Cons}$$

$$\frac{\Gamma \mid \Phi \vdash \mathsf{P}_1 \sqsubseteq \mathsf{P}_2 \qquad \Gamma \mid \Phi \vdash (\Delta).[\mathsf{P}_2]\ \mathsf{e}\ [\mathsf{Q}_2] \qquad \Gamma \mid \Phi \vdash \mathsf{Q}_2 \sqsubseteq \mathsf{Q}_1}{\Gamma \mid \Phi \vdash (\Delta).[\mathsf{P}_1]\ \mathsf{e}\ [\mathsf{Q}_1]}\ \textsc{Cons}$$

Other Rules for Commands

$$\frac{\begin{array}{c}\Gamma, \bar{\mathsf{y}}, \mathbf{this} \mid \Phi \vdash \mathsf{stable}(\mathsf{P}) \land \mathsf{stable}(\mathsf{Q}) \\ \Gamma \mid \Phi \vdash \triangleright(\mathsf{C::m} : (\bar{\mathsf{y}}).\ \{\mathsf{P}\}\ \{\mathsf{Q}\}) \qquad \Gamma \vdash \mathsf{C} : \mathsf{Class}\end{array}}{\Gamma \mid \Phi(\Delta).[\mathsf{P}[\bar{\mathsf{v}}/\bar{\mathsf{y}}, \mathsf{x}/\mathbf{this}] * \mathsf{x} : \mathsf{C}]\ \mathsf{x.m}(\bar{\mathsf{v}})\ [\mathsf{Q}[\bar{\mathsf{v}}/\bar{\mathsf{y}}, \mathsf{x}/\mathbf{this}]]}\ \textsc{Call}$$

$$\frac{\begin{array}{c}\Gamma, \mathbf{this} \mid \Phi \vdash \mathsf{stable}(\mathsf{P}) \land \mathsf{stable}(\mathsf{Q}) \\ \Gamma \mid \Phi \vdash \triangleright(\mathsf{C::m} : (-).\ \{\ulcorner \mathsf{P} \urcorner\}\ \{\mathsf{Q}\}) \qquad \Gamma \vdash \mathsf{C} : \mathsf{Class}\end{array}}{\Gamma \mid \Phi \vdash (\Delta).[\ulcorner \mathsf{P}[\mathsf{x}/\mathbf{this}] \urcorner * \mathsf{y} : \mathsf{C}]\ \mathbf{fork}(\mathsf{y.m})\ [\top]}\ \textsc{Fork}$$

$$\frac{\Gamma, \bar{\mathsf{x}} \mid \Phi \vdash \mathsf{stable}(\mathsf{P}) \land \mathsf{stable}(\mathsf{Q}) \qquad \Gamma \mid \Phi \vdash \triangleright(\mathsf{C} : (\bar{\mathsf{x}}).\ \{\mathsf{P}\}\ \{\mathsf{Q}\}) \qquad \Gamma \vdash \mathsf{C} : \mathsf{Class}}{\Gamma \mid \Phi(\Delta).[\mathsf{P}[\bar{\mathsf{v}}/\bar{\mathsf{x}}]]\ \mathbf{new}\ \mathsf{C}(\bar{\mathsf{v}})\ [\mathsf{Q}[\bar{\mathsf{v}}/\bar{\mathsf{x}}]]}\ \textsc{New}$$

$$\frac{\Gamma \mid \Phi \vdash (\Delta).[\mathsf{P}]\ \mathsf{e}_1\ [\mathsf{r}.\ \mathsf{Q}(\mathsf{r})] \qquad \Gamma \mid \Phi \vdash (\Delta, \mathsf{x}).[\mathsf{Q}(\mathsf{x})]\ \mathsf{e}_2\ [\mathsf{r}.\ \mathsf{R}(\mathsf{r})]}{\Gamma \mid \Phi \vdash (\Delta).[\mathsf{P}]\ \mathbf{let}\ \mathsf{x} = \mathsf{e}_1\ \mathbf{in}\ \mathsf{e}_2\ [\mathsf{r}.\ \mathsf{R}(\mathsf{r})]}\ \textsc{Bind}$$

$$\frac{\Gamma; - \vdash \mathsf{v} : \mathsf{Val}}{\Gamma \mid \Phi \vdash [\top]\ \mathsf{v}\ [\mathsf{r}.\ \mathsf{r} = \mathsf{v}]}\ \textsc{Val}$$

View-Shifts

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

$$\frac{}{\Gamma \mid \Phi \vdash \mathsf{x_f} \overset{1}{\mapsto} \mathsf{v} \sqsubseteq \mathsf{x_f} \overset{1}{\mapsto} \mathsf{v}'}\ \textsc{VPUpdate}$$

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

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

Derived Rules – Hoare triples for the SC Logic

$$\frac{\Gamma; - \vdash \mathsf{v} : \mathsf{Val} \qquad \mathsf{x} \in \Delta}{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{x.f} \mapsto \mathsf{v}\}\ \mathsf{x.f}\ \{\mathsf{r}.\ \mathsf{r} = \mathsf{v} * \mathsf{x.f} \mapsto \mathsf{v}\}}\ \textsc{S-Read}$$

$$\frac{\Gamma; - \vdash \mathsf{v} : \mathsf{Val} \qquad \mathsf{x} \in \Delta}{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{x.f} \mapsto _\}\ \mathsf{x.f} := \mathsf{v}\ \{\mathsf{r}.\ \mathsf{x.f} \mapsto \mathsf{v}\}}\ \textsc{S-Write}$$

$$\frac{\begin{array}{c}\Gamma, \bar{\mathsf{y}}, \mathbf{this}, \mathsf{r} \mid \Phi \vdash \mathsf{r}\text{-}\mathsf{stable}(\mathsf{P}) \wedge \mathsf{r}\text{-}\mathsf{stable}(\mathsf{Q}) \\ \Gamma \mid \Phi \vdash \triangleright(\mathsf{C::m} : (\bar{\mathsf{y}}).\ [\mathsf{P}]\ [\mathsf{r}.\ \mathsf{Q}]) \qquad \Gamma \vdash \mathsf{C} : \mathsf{Class}\end{array}}{\Gamma \mid \Phi(\Delta).\{\mathsf{P}[\bar{\mathsf{v}}/\bar{\mathsf{y}}, \mathsf{x}/\mathbf{this}] * \mathsf{x} : \mathsf{C}\}\ \mathsf{x.m}(\bar{\mathsf{v}})\ \{\mathsf{r}.\ \mathsf{Q}[\bar{\mathsf{v}}/\bar{\mathsf{y}}, \mathsf{x}/\mathbf{this}]\}}\ \textsc{S-Call}$$

$$\frac{\begin{array}{c}\Gamma, \bar{\mathsf{x}}, \mathsf{r} \mid \Phi \vdash \mathsf{stable}(\mathsf{P}) \wedge \mathsf{stable}(\mathsf{Q}) \\ \Gamma \mid \Phi \vdash \triangleright(\mathsf{C} : (\bar{\mathsf{x}}).\ [\mathsf{P}]\ [\mathsf{r}.\ \mathsf{Q}]) \qquad \Gamma \vdash \mathsf{C} : \mathsf{Class}\end{array}}{\Gamma \mid \Phi(\Delta).\{\mathsf{P}[\bar{\mathsf{v}}/\bar{\mathsf{x}}]\}\ \mathbf{new}\ \mathsf{C}(\bar{\mathsf{v}})\ \{\mathsf{r}.\ \mathsf{Q}[\bar{\mathsf{v}}/\bar{\mathsf{x}}]\}}\ \textsc{S-New}$$

$$\frac{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}\}\ \mathsf{e}\ \{\mathsf{r}.\ \mathsf{Q}\} \qquad \Gamma, \Delta \mid \Phi \vdash \mathsf{r}\text{-}\mathsf{stable}(\mathsf{R})}{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P} * \mathsf{R}\}\ \mathsf{e}\ \{\mathsf{r}.\ \mathsf{Q} * \mathsf{R}\}}\ \textsc{S-Frame}$$

$$\frac{\Gamma; \Delta \mid \Phi \mid \mathsf{P}_1 \vdash_{SC} \mathsf{P}_2 \qquad \Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}_2\}\ \mathsf{e}\ \{\mathsf{r}.\ \mathsf{Q}_2\} \qquad \Gamma, \mathsf{r}; \Delta \mid \Phi \mid \mathsf{Q}_2 \vdash_{SC} \mathsf{Q}_1}{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}_1\}\ \mathsf{e}\ \{\mathsf{r}.\ \mathsf{Q}_1\}}\ \textsc{S-Cons}$$

$$\frac{\Gamma; \Delta \mid \Phi \mid \mathsf{P} \vdash \mathsf{v} = \mathbf{true} \qquad \Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}\}\ \mathsf{e}_1\ \{\mathsf{r}.\ \mathsf{Q}\}}{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}\}\ \mathbf{if}\ \mathsf{v}\ \mathbf{then}\ \mathsf{e}_1\ \mathbf{else}\ \mathsf{e}_2\ \{\mathsf{r}.\ \mathsf{Q}\}}\ \textsc{S-If-T}$$

$$\frac{\Gamma; \Delta \mid \Phi \mid \mathsf{P} \vdash \mathsf{v} = \mathbf{false} \qquad \Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}\}\ \mathsf{e}_2\ \{\mathsf{r}.\ \mathsf{Q}\}}{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}\}\ \mathbf{if}\ \mathsf{v}\ \mathbf{then}\ \mathsf{e}_1\ \mathbf{else}\ \mathsf{e}_2\ \{\mathsf{r}.\ \mathsf{Q}\}}\ \textsc{S-If-F}$$

$$\frac{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}\}\ \mathsf{e}_1\ \{\mathsf{r}.\ \mathsf{Q}(\mathsf{r})\} \qquad \Gamma \mid \Phi \vdash (\Delta, \mathsf{x}).\{\mathsf{Q}(\mathsf{x})\}\ \mathsf{e}_2\ \{\mathsf{r}.\ \mathsf{R}(\mathsf{r})\}}{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}\}\ \mathbf{let}\ \mathsf{x} = \mathsf{e}_1\ \mathbf{in}\ \mathsf{e}_2\ \{\mathsf{r}.\ \mathsf{R}(\mathsf{r})\}}\ \textsc{S-Bind}$$

$$\frac{\Gamma; - \vdash \mathsf{v} : \mathsf{Val}}{\Gamma \mid \Phi \vdash \{\top\}\ \mathsf{v}\ \{\mathsf{r}.\ \mathsf{r} = \mathsf{v}\}}\ \textsc{S-Val} \qquad \frac{\Gamma \mid \Phi \vdash (\Delta).[\overline{\mathsf{P}}]\ \mathsf{e}\ [\overline{\mathsf{Q}}]}{\Gamma \mid \Phi \vdash (\Delta).\{\mathsf{P}\}\ \mathsf{e}\ \{\mathsf{r}.\ \mathsf{Q}(\mathsf{r})\}}\ \textsc{S-Shift}$$

3.4.2 Later operator

In the following proof rules, the rules for distribution over quantifiers and binary connectives work for both SC- and TSO-level connectives (except for $\mathcal{U}$ operator, which is only defined on the TSO level), hence we omit the subscripts on the entailments.

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

$$\frac{op \in \{\wedge, \vee, *, \mathcal{U}_t\}}{\Gamma; \Delta \mid \Phi \mid \triangleright(\mathsf{P}\ op\ \mathsf{Q}) \dashv\vdash (\triangleright \mathsf{P})\ op\ (\triangleright \mathsf{Q})}\ \textsc{LBin} \qquad \frac{E \in \{\ulcorner - \urcorner, \ulcorner - \textbf{ in } t \urcorner\}}{\Gamma; \Delta \mid \Phi \mid \triangleright(E(\mathsf{P})) \dashv\vdash E(\triangleright P)}\ \textsc{LEmb}$$

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

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

3.4.3 Reasoning about stability

Stability consists of two components: stability under region interference ($\mathsf{r\text{-}stable}$) and stability under store-buffer interference ($\mathsf{b\text{-}stable}$). The first of these is defined for both level of assertions, the second only for $\mathsf{Prop}_{\mathsf{TSO}}$. Note how both embeddings preserve stability under regions and *grant* stability under store-buffer interference, and that the explicit stabilization does indeed provide stability.

$$\frac{\Gamma \mid \Phi \vdash \forall \alpha \notin A.\ \forall x \in X.\ T(\alpha)(x) \subseteq X}{\Gamma \mid \Phi \vdash \mathsf{r\text{-}stable}(\mathsf{region}(X, T, n) * \circledast_{\alpha \in A} [\alpha]_1^n)}$$

$$\frac{\Gamma \vdash \mathsf{I} : \mathsf{AId} \to \mathcal{P}(\mathsf{SId} \times \mathsf{SId}) \qquad \Gamma \vdash \mathsf{n} : \mathsf{RId}}{\Gamma \mid \Phi \vdash \mathsf{r\text{-}stable}(\mathsf{rintr}(\mathsf{I}, \mathsf{n}))}$$

$$\frac{c \in \{\top, \bot, \mathsf{emp}\}}{\Gamma \mid \Phi \vdash \mathsf{stable}(c)} \qquad \frac{}{\Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{x.f} \mapsto \mathsf{v})} \qquad \frac{}{\Gamma \mid \Phi \vdash \mathsf{stable}(\mathsf{x} : \mathsf{C})}$$

$$\frac{}{\Gamma \mid \Phi \vdash \mathsf{stable}(\llparenthesis \mathsf{P} \rrparenthesis)} \qquad \frac{}{\Gamma \mid \Phi \vdash \mathsf{b\text{-}stable}(\ulcorner \mathsf{P} \urcorner)} \qquad \frac{}{\Gamma \mid \Phi \vdash \mathsf{b\text{-}stable}(\ulcorner \mathsf{P} \textbf{ in } \mathsf{t} \urcorner)}$$

$$\frac{\Gamma \mid \Phi \vdash \mathsf{r\text{-}stable}(\mathsf{P})}{\Gamma \mid \Phi \vdash \mathsf{r\text{-}stable}(\ulcorner \mathsf{P} \urcorner)} \qquad \frac{\Gamma \mid \Phi \vdash \mathsf{r\text{-}stable}(\mathsf{P})}{\Gamma \mid \Phi \vdash \mathsf{r\text{-}stable}(\ulcorner \mathsf{P} \textbf{ in } \mathsf{t} \urcorner)}$$

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

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

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

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

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

3.4.4 Method verification

To verify a method or a constructor, we verify its body:

$$\frac{\begin{array}{cc}\Gamma; \Delta, \mathbf{this} \vdash \mathsf{P} : \mathsf{TId} \to \mathsf{Prop_{TSO}} & \Gamma; \Delta, \mathbf{this} \vdash \mathsf{Q} : \mathsf{TId} \to \mathsf{Val} \to \mathsf{Prop_{TSO}} \\ \mathsf{body}(\mathsf{C}, \mathsf{m}) = (\mathsf{T\ m}(\Delta) = \mathsf{e}) & \Gamma \mid \Phi \vdash (\Delta, \mathbf{this}).[\mathsf{P} * \mathbf{this} : \mathsf{C}]\ \mathsf{e}\ [\mathsf{Q}]\end{array}}{\Gamma \mid \Phi \vdash \mathsf{C}{::}\mathsf{m} : (\Delta).\ \{\mathsf{P}\}\ \{\mathsf{Q}\}}$$

$$\frac{\begin{array}{c}\mathsf{ctorBody}(\mathsf{C}) = (\mathsf{C}(\Delta) = \mathsf{e}) \qquad \mathsf{FV} \in \mathcal{P}_{fin}(\Delta(FName \times PVal)) \\ \Gamma; \Delta \vdash \mathsf{P} : \mathsf{TId} \to \mathsf{Prop_{TSO}} \qquad \Gamma; \Delta \vdash \mathsf{Q} : \mathsf{TId} \to \mathsf{Val} \to \mathsf{Prop_{TSO}} \\ \Gamma \mid \Phi \vdash (\Delta, \mathbf{this}).[\mathsf{P} * \circledast_{\mathsf{f} \in \mathsf{fields}(C)} \ulcorner \mathbf{this}.\mathsf{f} \mapsto \mathbf{null} \urcorner * \circledast_{(\mathsf{f},\mathsf{v}) \in \mathsf{FV}} \mathbf{this}_{\mathsf{f}} \overset{1}{\mapsto} \mathsf{v} * \mathbf{this} : \mathsf{C}]\ \mathsf{e}\ [\mathsf{Q}]\end{array}}{\Gamma \mid \Phi \vdash \mathsf{C} : (\Delta).\ \{\mathsf{P}\}\ \{\mathsf{Q}\}}$$

Note that constructors are syntactically required to return the value of the newly allocated object. At the beginning of the constructor verification, one can additionally choose to allocate any finite number of phantom fields for that object, with arbitrary values.

4 Interpretation

4.1 Types

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

4.2 Contexts

$$
\begin{aligned}
[\![\Gamma, \mathsf{x} : \tau]\!] &= [\![\Gamma]\!] \times [\![\vdash \tau : \mathsf{Type}]\!] & \qquad [\![\varepsilon]\!] &= 1 \\
[\![\Delta, \mathsf{x} : \mathsf{Val}]\!] &= [\![\Delta]\!] \times [\![\vdash \mathsf{Val} : \mathsf{Type}]\!] & \qquad [\![\varepsilon]\!] &= 1
\end{aligned}
$$

4.3 Lambda calculus

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

4.4 TSO Assertion Logic

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

$$[\![\Gamma;\Delta \vdash \ulcorner \mathsf{P} \urcorner : \mathsf{Prop_{TSO}}]\!](\gamma,\delta) \stackrel{\text{def}}{=} \text{let } p = [\![\Gamma;\Delta \vdash \mathsf{P} : \mathsf{Prop_{SC}}]\!](\gamma,\delta) \text{ in } \ulcorner p \urcorner$$

$$[\![\Gamma;\Delta \vdash \ulcorner \mathsf{P} \textbf{ in } t \urcorner : \mathsf{Prop_{TSO}}]\!](\gamma,\delta) \stackrel{\text{def}}{=} \text{let } p = [\![\Gamma;\Delta \vdash \mathsf{P} : \mathsf{Prop_{SC}}]\!](\gamma,\delta) \text{ in } \ulcorner p \textbf{ in } t \urcorner$$

$$\begin{aligned}[\![\Gamma;\Delta \vdash \mathsf{P}\,\mathcal{U}_t\,\mathsf{Q} : \mathsf{Prop_{TSO}}]\!](\gamma,\delta) \stackrel{\text{def}}{=} \; & \text{let } p = [\![\Gamma;\Delta \vdash \mathsf{P} : \mathsf{Prop_{TSO}}]\!](\gamma,\delta) \text{ in} \\ & \text{let } q = [\![\Gamma;\Delta \vdash \mathsf{Q} : \mathsf{Prop_{TSO}}]\!](\gamma,\delta) \text{ in} \\ & \quad p\,\mathcal{U}_t\,q\end{aligned}$$

$$[\![\Gamma;\Delta \vdash (\!|P|\!) : \mathsf{Prop_{TSO}}]\!](\gamma,\delta) \stackrel{\text{def}}{=} \text{let } p = [\![\Gamma;\Delta \vdash P : \mathsf{Prop_{TSO}}]\!](\gamma,\delta) \text{ in } \mathrm{cl}(p)$$

$$[\![\Gamma;\Delta \vdash \mathsf{emp} : \mathsf{Prop_{TSO}}]\!](\gamma,\delta) = \mathcal{M}$$

$$[\![\Gamma;\Delta \vdash \mathsf{P} * \mathsf{Q} : \mathsf{Prop_{TSO}}]\!](\gamma,\delta) = [\![\Gamma;\Delta \vdash \mathsf{P} : \mathsf{Prop_{TSO}}]\!](\gamma,\delta) * [\![\Gamma;\Delta \vdash \mathsf{Q} : \mathsf{Prop_{TSO}}]\!](\gamma,\delta)$$

4.5 SC Assertion Logic

$$[\![\Gamma;\Delta \vdash \bot : \mathsf{Prop_{SC}}]\!](\gamma,\delta) = \emptyset$$

$$[\![\Gamma;\Delta \vdash \top : \mathsf{Prop_{SC}}]\!](\gamma,\delta) = \mathcal{H}$$

$$[\![\Gamma;\Delta \vdash \mathsf{P} \wedge \mathsf{Q} : \mathsf{Prop_{SC}}]\!](\gamma,\delta) = [\![\Gamma;\Delta \vdash \mathsf{P} : \mathsf{Prop_{SC}}]\!](\gamma,\delta) \cap [\![\Gamma;\Delta \vdash \mathsf{Q} : \mathsf{Prop_{SC}}]\!](\gamma,\delta)$$

$$[\![\Gamma;\Delta \vdash \mathsf{P} \vee \mathsf{Q} : \mathsf{Prop_{SC}}]\!](\gamma,\delta) = [\![\Gamma;\Delta \vdash \mathsf{P} : \mathsf{Prop_{SC}}]\!](\gamma,\delta) \cup [\![\Gamma;\Delta \vdash \mathsf{Q} : \mathsf{Prop_{SC}}]\!](\gamma,\delta)$$

$$\begin{aligned}[\![\Gamma;\Delta \vdash \mathsf{P} \Rightarrow \mathsf{Q} : \mathsf{Prop_{SC}}]\!](\gamma,\delta) = \; & \{m \in \mathcal{H} \mid \forall n \geq m. \\ & \quad n \in [\![\Gamma;\Delta \vdash \mathsf{P} : \mathsf{Prop_{SC}}]\!](\gamma,\delta) \;\Rightarrow \\ & \quad n \in [\![\Gamma;\Delta \vdash \mathsf{Q} : \mathsf{Prop_{SC}}]\!](\gamma,\delta)\}\end{aligned}$$

$$[\![\Gamma;\Delta \vdash \forall \mathsf{x} : \tau.\ \mathsf{P} : \mathsf{Prop_{SC}}]\!](\gamma,\delta) = \bigcap_{v \in [\![\vdash \tau : \mathsf{Type}]\!]} [\![\Gamma, \mathsf{x} : \tau;\Delta \vdash \mathsf{P} : \mathsf{Prop_{SC}}]\!]((\gamma,v),\delta)$$

$$[\![\Gamma;\Delta \vdash \exists \mathsf{x} : \tau.\ \mathsf{P} : \mathsf{Prop_{SC}}]\!](\gamma,\delta) = \bigcup_{v \in [\![\vdash \tau : \mathsf{Type}]\!]} [\![\Gamma, \mathsf{x} : \tau;\Delta \vdash \mathsf{P} : \mathsf{Prop_{SC}}]\!]((\gamma,v),\delta)$$

$$[\![\Gamma;\Delta \vdash \mathsf{emp} : \mathsf{Prop_{SC}}]\!](\gamma,\delta) = \mathcal{H}$$

$$[\![\Gamma;\Delta \vdash \mathsf{P} * \mathsf{Q} : \mathsf{Prop_{SC}}]\!](\gamma,\delta) = [\![\Gamma;\Delta \vdash \mathsf{P} : \mathsf{Prop_{SC}}]\!](\gamma,\delta) * [\![\Gamma;\Delta \vdash \mathsf{Q} : \mathsf{Prop_{SC}}]\!](\gamma,\delta)$$

$$\begin{aligned}[\![\Gamma;\Delta \vdash \mathsf{x.f} \mapsto \mathsf{v} : \mathsf{Prop_{SC}}]\!](\gamma,\delta) = \; & \text{let } o = [\![\Gamma;\Delta \vdash \mathsf{x} : \mathsf{Val}]\!](\gamma,\delta) \text{ in} \\ & \text{let } f = [\![\Gamma;\Delta \vdash \mathsf{f} : \mathsf{Field}]\!](\gamma,\delta) \text{ in} \\ & \text{let } v = [\![\Gamma;\Delta \vdash \mathsf{v} : \mathsf{Val}]\!](\gamma,\delta) \text{ in} \\ & \quad \{m \in \mathcal{H} \mid m.l(o,f) = v\}\end{aligned}$$

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

$$[\![\Gamma;\Delta \vdash \mathsf{rintr}(\mathsf{I},\mathsf{R}) : \mathsf{Prop}_{\mathsf{SC}}]\!](\gamma,\delta) = rintr([\![\Gamma;\Delta \vdash \mathsf{I} : \mathsf{SId} \to \mathsf{Prop}_{\mathsf{TSO}}]\!](\gamma,\delta), [\![\Gamma;\Delta \vdash \mathsf{R} : \mathsf{RId}]\!](\gamma,\delta))$$

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

where ϕ embeds predicates from our set-theoretic metalogic in the topos of trees. For more details, consult the iCAP technical report [6].

4.6 Specification Logic

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

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

$$\begin{aligned}[\![\Gamma \vdash (\Delta).[\mathsf{P}]\ \mathsf{e}\ [\mathsf{Q}]]\!](\gamma) \stackrel{\text{def}}{=} & \forall t \in TId.\ \forall \delta \in [\![\Delta]\!]. \\ & \text{let } p = [\![\Gamma;\Delta \vdash \mathsf{P} : \mathsf{TId} \to \mathsf{Prop}_{\mathsf{TSO}}]\!](\gamma,\delta) \text{ in} \\ & \text{let } q = [\![\Gamma;\Delta \vdash \mathsf{Q} : \mathsf{TId} \to \mathsf{Val} \to \mathsf{Prop}_{\mathsf{TSO}}]\!](\gamma,\delta) \text{ in} \\ & \quad \text{safe}((t, \delta(\mathsf{e})), p(t), q(t)) \\ [\![\Gamma \vdash (\Delta).\langle\mathsf{P}\rangle\ \mathsf{e}\ \langle\mathsf{Q}\rangle^A]\!](\gamma) \stackrel{\text{def}}{=} & \forall t \in TId.\ \forall \delta \in [\![\Delta]\!].\ \forall a \in Act.\ \forall v \in Val. \\ & \text{let } p = [\![\Gamma;\Delta \vdash \mathsf{P} : \mathsf{TId} \to \mathsf{Prop}_{\mathsf{TSO}}]\!](\gamma,\delta) \text{ in} \\ & \text{let } q = [\![\Gamma;\Delta \vdash \mathsf{Q} : \mathsf{TId} \to \mathsf{Val} \to \mathsf{Prop}_{\mathsf{TSO}}]\!](\gamma,\delta) \text{ in} \\ & \quad (t, \delta(e)) \xrightarrow{a} \{(t, v)\} \Rightarrow a \text{ sat}_A\ \{p(t)\}\,\{q(t)(v)\}\end{aligned}$$

$$\begin{aligned}[\![\Gamma \vdash \mathsf{r\text{-}stable}(\mathsf{P}) : \mathsf{Spec}]\!](\gamma) &= \text{let } p = [\![\Gamma;\varepsilon \vdash \mathsf{P} : \mathsf{Prop}_{\mathsf{SC}}]\!](\gamma,\varepsilon) \text{ in rstable}(p) \\ [\![\Gamma \vdash \mathsf{b\text{-}stable}(\mathsf{P}) : \mathsf{Spec}]\!](\gamma) &= \text{let } p = [\![\Gamma;\varepsilon \vdash \mathsf{P} : \mathsf{Prop}_{\mathsf{TSO}}]\!](\gamma,\varepsilon) \text{ in bstable}(p) \\ [\![\Gamma \vdash \mathsf{P} \sqsubseteq^{\mathsf{R}} \mathsf{Q} : \mathsf{Spec}]\!](\gamma) &= \text{let } p = [\![\Gamma;\varepsilon \vdash \mathsf{P} : \mathsf{Prop}_{\mathsf{TSO}}]\!](\gamma,\varepsilon) \text{ in} \\ & \quad\ \text{let } q = [\![\Gamma;\varepsilon \vdash \mathsf{Q} : \mathsf{Prop}_{\mathsf{TSO}}]\!](\gamma,\varepsilon) \text{ in} \\ & \quad\ \text{let } R = [\![\Gamma \vdash \mathsf{R} : \mathcal{P}(\mathsf{RId})]\!](\gamma) \text{ in} \\ & \quad\quad p \sqsubseteq_R q\end{aligned}$$

4.7 Read and Flush Judgments

$$\begin{aligned}
[\![\Gamma;\Delta \mid \mathsf{P} \vdash_{\mathrm{fl}(t)} \mathsf{Q}]\!](\gamma,\delta) = {} & \text{let } p = [\![\Gamma;\Delta \vdash \mathsf{P} : \mathsf{Prop_{TSO}}]\!](\gamma,\delta) \text{ in} \\
& \text{let } q = [\![\Gamma;\Delta \vdash \mathsf{Q} : \mathsf{Prop_{TSO}}]\!](\gamma,\delta) \text{ in} \\
& \quad p \vdash_{\mathrm{fl}(t)} q \\
[\![\Gamma;\Delta \mid \mathsf{P} \vdash_{\mathrm{rd}(t)} \mathsf{x.f} \mapsto \mathsf{v}]\!](\gamma,\delta) = {} & \text{let } p = [\![\Gamma;\Delta \vdash \mathsf{P} : \mathsf{Prop_{TSO}}]\!](\gamma,\delta) \text{ in} \\
& \text{let } o = [\![\Gamma;\Delta \vdash \mathsf{x} : \mathsf{Val}]\!](\gamma,\delta) \text{ in} \\
& \text{let } v = [\![\Gamma;\Delta \vdash \mathsf{v} : \mathsf{Val}]\!](\gamma,\delta) \text{ in} \\
& \quad p \vdash_{\mathrm{rd}(t)} o.\mathsf{f} \mapsto v
\end{aligned}$$

4.8 Embeddings and later operators

$$\begin{aligned}
[\![\Gamma \vdash \mathsf{valid(P)} : \mathsf{Spec}]\!](\gamma) &= \mathrm{valid}([\![\Gamma; - \vdash \mathsf{P} : \mathsf{Prop_{TSO}}]\!](\gamma)) \\
[\![\Gamma \vdash \triangleright \mathsf{S} : \mathsf{Spec}]\!](\gamma) &= \triangleright([\![\Gamma \vdash \mathsf{S} : \mathsf{Spec}]\!](\gamma))
\end{aligned}$$

$$\begin{aligned}
[\![\Gamma;\Delta \vdash \mathsf{spec(S)} : \mathsf{Prop_{SC}}]\!](\gamma,\delta) &= spec([\![\Gamma \vdash \mathsf{S} : \mathsf{Spec}]\!](\gamma)) \\
[\![\Gamma;\Delta \vdash \triangleright \mathsf{P} : \mathsf{Prop_{TSO}}]\!](\gamma,\delta) &= \triangleright([\![\Gamma;\Delta \vdash \mathsf{P} : \mathsf{Prop_{TSO}}]\!](\gamma,\delta)) \\
[\![\Gamma;\Delta \vdash \triangleright \mathsf{P} : \mathsf{Prop_{SC}}]\!](\gamma,\delta) &= \triangleright([\![\Gamma;\Delta \vdash \mathsf{P} : \mathsf{Prop_{SC}}]\!](\gamma,\delta))
\end{aligned}$$

4.9 Entailment

$$\begin{aligned}
&[\![\Gamma;\Delta \mid \Phi \mid \mathsf{P} \vdash_{TSO} \mathsf{Q}]\!] = \\
&\quad \forall \gamma \in [\![\Gamma]\!].\ \forall \delta \in [\![\Delta]\!].\ [\![\Phi]\!](\gamma) \Rightarrow [\![\Gamma;\Delta \vdash \mathsf{P} : \mathsf{Prop_{TSO}}]\!](\gamma,\delta) \subseteq [\![\Gamma;\Delta \vdash \mathsf{Q} : \mathsf{Prop_{TSO}}]\!](\gamma,\delta)
\end{aligned}$$

$$\begin{aligned}
&[\![\Gamma;\Delta \mid \Phi \mid \mathsf{P} \vdash_{SC} \mathsf{Q}]\!] = \\
&\quad \forall \gamma \in [\![\Gamma]\!].\ \forall \delta \in [\![\Delta]\!].\ [\![\Phi]\!](\gamma) \Rightarrow [\![\Gamma;\Delta \vdash \mathsf{P} : \mathsf{Prop_{SC}}]\!](\gamma,\delta) \subseteq [\![\Gamma;\Delta \vdash \mathsf{Q} : \mathsf{Prop_{SC}}]\!](\gamma,\delta)
\end{aligned}$$

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

5 Soundness

Lemma 32.

$$\forall p \in Prop_{SC}.\ \triangleright \ulcorner p \urcorner = \ulcorner \triangleright p \urcorner$$

Proof ($\subseteq$). Assume $\triangleright(l, s, U, \zeta) \in \ulcorner p \urcorner$. Then by definition of the embedding,

$$\triangleright(\exists l' \in \Delta(\mathit{LState}).\ l' \leq l \wedge (l', s, U, \zeta) \in p \wedge \mathrm{lfd}(l', U))$$

Since later commutes over existentials over constant sets and conjunctions and $l' \leq l$ and lfd simply reduces to (universally quantified) equalities on constant sets, which are upwards-closed, the above is equivalent to:

$$\exists l' \in \Delta(\mathit{LState}).\ (l' \leq l \vee \triangleright\bot) \wedge (l', s, U, \zeta) \in \triangleright p \wedge (\mathrm{lfd}(l', U) \vee \triangleright\bot))$$

In case $\triangleright\bot$ then $(l, s, U, \zeta) \in \ulcorner \triangleright p \urcorner$ holds trivially, by taking l' to be the empty local state, which is trivially smaller than l. Otherwise, the conclusion follows directly from the assumptions. □

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

Lemma 33.

$$\forall p, q \in \mathit{Prop}_{TSO}.\ \triangleright(p\, \mathcal{U}_t\, q) = (\triangleright p)\, \mathcal{U}_t\, (\triangleright q)$$

Proof ($\subseteq$). Assume $(l, s, U, \zeta) \in \triangleright(p\, \mathcal{U}_t\, q)$. Hence, by the definition of $\mathcal{U}_t$

$$\begin{aligned}
\triangleright(\ &\exists \alpha, \beta \in \Delta(\mathit{SBuffer}).\ \exists o \in \Delta(\mathit{OId}).\ \exists f \in \Delta(\mathit{FName}).\ \exists v \in \Delta(\mathit{Val}).\\
&\quad U(t) = \alpha \cdot (o, f, v) \cdot \beta \wedge (l, s, U[t \mapsto \alpha], \zeta) \in p \wedge\\
&\quad (\mathrm{flush}(l, \alpha \cdot (o, f, v)), s, U[t \mapsto \beta], \zeta) \in q)
\end{aligned}$$

Since constant sets are trivially total and $\triangleright$ commutes over $\wedge$ this reduces to

$$\begin{aligned}
&\exists \alpha, \beta \in \Delta(\mathit{SBuffer}).\ \exists o \in \Delta(\mathit{OId}).\ \exists f \in \Delta(\mathit{FName}).\ \exists v \in \Delta(\mathit{Val}).\\
&\quad \triangleright(U(t) = \alpha \cdot (o, f, v) \cdot \beta) \wedge (l, s, U[t \mapsto \alpha], \zeta) \in \triangleright p \wedge\\
&\quad (\mathrm{flush}(l, \alpha \cdot (o, f, v)), s, U[t \mapsto \beta], \zeta) \in \triangleright q
\end{aligned}$$

Since equality on constant sets is upwards-closed, this is equivalent to

$$\begin{aligned}
&\exists \alpha, \beta \in \Delta(\mathit{SBuffer}).\ \exists o \in \Delta(\mathit{OId}).\ \exists f \in \Delta(\mathit{FName}).\ \exists v \in \Delta(\mathit{Val}).\\
&\quad U(t) = \alpha \cdot (o, f, v) \cdot \beta \wedge (l, s, U[t \mapsto \alpha], \zeta) \in \triangleright p \wedge\\
&\quad (\mathrm{flush}(l, \alpha \cdot (o, f, v)), s, U[t \mapsto \beta], \zeta) \in \triangleright q
\end{aligned}$$

Hence $(l, s, U, \zeta) \in (\triangleright p)\, \mathcal{U}_t\, (\triangleright q)$. □

Proof ($\supseteq$). Assume $(l, s, U, \zeta) \in (\triangleright p)\, \mathcal{U}_t\, (\triangleright q)$. Then

$$\begin{aligned}
&\exists \alpha, \beta \in \Delta(\mathit{SBuffer}).\ \exists o \in \Delta(\mathit{OId}).\ \exists f \in \Delta(\mathit{FName}).\ \exists v \in \Delta(\mathit{Val}).\\
&\quad U(t) = \alpha \cdot (o, f, v) \cdot \beta \wedge (l, s, U[t \mapsto \alpha], \zeta) \in \triangleright p \wedge\\
&\quad (\mathrm{flush}(l, \alpha \cdot (o, f, v)), s, U[t \mapsto \beta], \zeta) \in \triangleright q
\end{aligned}$$

by monotonicity of $\triangleright$ and by commuting $\triangleright$ over $\wedge$ and $\exists$, it follows that

$$\triangleright(\exists \alpha, \beta \in \Delta(\mathit{SBuffer}).\ \exists o \in \Delta(\mathit{OId}).\ \exists f \in \Delta(\mathit{FName}).\ \exists v \in \Delta(\mathit{Val}).\\ U(t) = \alpha \cdot (o, f, v) \cdot \beta \wedge (l, s, U[t \mapsto \alpha], \zeta) \in p \wedge\\ (\mathrm{flush}(l, \alpha \cdot (o, f, v)), s, U[t \mapsto \beta], \zeta) \in q)$$

Thus $(l, s, U, \zeta) \in \triangleright(p\, \mathcal{U}_t\, q)$. □

Lemma 34. *For any $A \in \mathcal{P}(\Delta(\mathit{RId}))$, $p \in \mathit{Prop}_{\mathit{TSO}}$, $f \in \Delta(\mathit{FName})$, $v_1, v_2 \in \Delta(\mathit{Val})$, $t \in \Delta(\mathit{TId})$ and $o \in \Delta(\mathit{OId})$,*

$$p \vdash_{\mathrm{rd}(t)} o.f \mapsto v_1 \Rightarrow \mathit{read}(t, o, f, v_2)\ \mathrm{sat}_A\ \{p\}\, \{p * v_1 = v_2\}$$

Proof. Let $r \in \mathit{Prop}_{\mathit{TSO}}$, $m \in \mathcal{M}$, $h \in \Delta(\mathit{Heap})$, and $U \in \Delta(\mathit{SPool})$ such that

$$m \in p * \triangleright r \qquad (h, U) \in \lfloor m \rfloor_A \qquad \underset{A}{\mathrm{stable}}(r).$$

Hence, there exists $m_1, m_2 \in \mathcal{M}$ such that $m = m_1 \bullet m_2$, $m_1 \in p$ and $m_2 \in \triangleright r$. If $t \notin \mathrm{dom}(U)$, $[\![\mathit{read}(t, o, f, v_2)]\!](h, U) = \emptyset$ and the conclusion holds vacuously. Assume, then, that $t \in \mathrm{dom}(U)$. Now, from the definition of $\vdash_{\mathrm{rd}(t)}$ it follows that

$$\triangleright \mathrm{lookup}(o.f, m_1.U(t), m_1.l) = v_1.$$

Since equalities on terms of constant sets are upwards-closed it follows that

$$\mathrm{lookup}(o.f, m_1.U(t), m_1.l) = v_1 \vee \triangleright\bot$$

The $\triangleright\bot$ case follows trivially as the definition of atomic satisfaction only requires the post-condition to hold later. In the $\mathrm{lookup}(o.f, m_1.U(t), m_1.l) = v_1$ case, $o.f \in \mathrm{dom}\, h$, and so $\lightning \notin [\![\mathit{read}(t, o, f, v_2)]\!](h, U)$. Now we can take any $(h', U') \in [\![\mathit{read}(t, o, f, v_2)]\!](h, U)$. By definition, it follows that $h = h'$, $U = U'$, and $v_1 = v_2$. We pick m' to be m and thus have to show that

$$\triangleright(m \in p * v_1 = v_1 * r) \qquad \triangleright((h, U) \in \lfloor m \rfloor_A)$$

Both follow easily from monotonicity of $\triangleright$ and the fact that $\varepsilon \in v_1 = v_1$. □

Lemma 35. *For any $A \in \mathcal{P}(\Delta(\mathit{RId}))$, $p, q \in \mathit{Prop}_{\mathit{TSO}}$, $f \in \Delta(\mathit{FName})$, $v_1, v_2 \in \Delta(\mathit{Val})$, $t \in \Delta(\mathit{TId})$ and $o \in \Delta(\mathit{OId})$, if* bstable q *then*

$$p \vdash_{\mathrm{fl}(t)} \ulcorner o.f \mapsto v_2 \urcorner * q \Rightarrow \mathit{write}(t, o, f, v_1)\ \mathrm{sat}_A\ \{p\}\, \{p\, \mathcal{U}_t\, (q * \ulcorner o.f \mapsto v_1 \urcorner)\}$$

Proof. Let $r \in \mathit{Prop}_{\mathit{TSO}}$, $m \in \mathcal{M}$, $h \in \Delta(\mathit{Heap})$ and $U \in \Delta(\mathit{SPool})$ such that

$$m \in p * \triangleright r \qquad (h, U) \in \lfloor m \rfloor_A \qquad \underset{A}{\mathrm{stable}}(r)$$

Hence, there exist $m_1, m_2 \in \mathcal{M}$ such that $m = m_1 \bullet m_2$, $m_1 \in p$ and $m_2 \in \triangleright r$. Let

$$m = (l, s, U, \zeta) \qquad m_1 = (l_1, s, U, \zeta) \qquad m_2 = (l_2, s, U, \zeta)$$

Analogously to the read case, we only need to consider the case when $t \in \mathrm{dom}(U)$. From the definition of $\vdash_{\mathrm{fl}(t)}$ it follows that

$$\triangleright(\mathrm{flush}(l_1, U(t)), s, U[t \mapsto \varepsilon], \zeta) \in \ulcorner o.f \mapsto v_2 \urcorner * q.$$

It follows that $\triangleright\mathrm{lookup}(o.f, \varepsilon, \mathrm{flush}(l_1, U(t))) = v_2$, and so $\triangleright\mathrm{lookup}(o.f, U(t), l_1) = v_2$. Since equalities on constant sets are upwwards-closed, it follows that

$$\mathrm{lookup}(o.f, U(t), l_1) = v_2 \vee \triangleright\bot$$

As the $\triangleright\bot$ case follows easily from Lemma 15, assume $\mathrm{lookup}(o.f, U(t), l_1) = v_2$. Then $(o, f) \in \mathrm{dom}(h)$ and thus $\lightning \notin [\![\mathit{write}(t, o, f, v_1)]\!](h, U)$. Now, take any $(h', U') \in [\![\mathit{write}(t, o, f, v_1)]\!](h, U)$. By definition of action semantics, this means that $h' = h$ and $U' = U[t \mapsto U(t) \cdot (o, f, v_1)]$.

Take $m' = (l, s, U', \zeta)$. Since $(o, f) \in \mathrm{dom}(l_1)$, $\triangleright((l_2, s, U', \zeta) \in r)$ and $(h, U') \in \lfloor m' \rfloor_A$ follow by stability of r and interpretations of the shared regions. Hence, it suffices to show that $\triangleright(l_1, s, U', \zeta) \in p\, \mathcal{U}_t\, (q * \ulcorner o.f \mapsto v_1 \urcorner)$. To show this, we pick the final update in $U'(t)$, and thus need to show

$$\triangleright(l_1, s, U, \zeta) \in p \qquad \triangleright(\mathrm{flush}(l_1, U(t) \cdot (o, f, v_1)), s, U[t \mapsto \varepsilon], \zeta) \in q * \ulcorner o.f \mapsto v_1 \urcorner$$

The first property follows from our assumptions and monotonicity of later; the second is slightly more involved. From the property obtained from $\vdash_{\mathrm{fl}(t)}$, we learn that there exist $l_3, l_4 \in \Delta(\mathit{LState})$, such that $\triangleright l_3 \bullet l_4 = \mathrm{flush}(l_1, U(t))$, $\triangleright(l_3, s, U[t \mapsto \varepsilon], \zeta) \in q$ and $\triangleright(l_4, s, U[t \mapsto \varepsilon], \zeta) \in \ulcorner o.f \mapsto v_2 \urcorner$. We can now show, using the properties of flush, that

$$\mathrm{flush}(l_1, U(t) \cdot (o, f, v_1)) = \mathrm{flush}(l_3, (o, f, v_1)) \bullet \mathrm{flush}(l_4, (o, f, v_1)).$$

Since $\triangleright(o, f) \in \mathrm{dom}(l_4)$ has to hold, we can also conclude easily that

$$\triangleright(\mathrm{flush}(l_4, (o, f, v_1)), s, U[t \mapsto \varepsilon], \zeta) \in \ulcorner o.f \mapsto v_1 \urcorner.$$

However, since $\triangleright(o, f) \notin \mathrm{dom}(l_3)$, $\triangleright(l_3, U[t \mapsto \varepsilon])\ R_{sb}\ (\mathrm{flush}(l_3, (o, f, v_1)), U[t \mapsto \varepsilon])$, which concludes the proof, since q is stable under store-buffer interference. ☐

Lemma 36. *For any $A \in \mathcal{P}(\Delta(\mathit{RId}))$, $p, q \in \mathit{Prop}_{\mathit{TSO}}$, $t \in \Delta(\mathit{TId})$,*

$$p \vdash_{\mathrm{fl}(t)} q \Rightarrow \mathit{fence}(t)\ \mathrm{sat}_A\ \{p\}\,\{q\}\,.$$

Proof. Let $r \in \mathit{Prop}_{\mathit{TSO}}$, $m \in \mathcal{M}$, $h \in \Delta(\mathit{Heap})$ and $U \in \Delta(\mathit{SPool})$ such that

$$m \in p * \triangleright r \qquad (h, U) \in \lfloor m \rfloor_A \qquad \underset{A}{\mathrm{stable}}(r)$$

Hence, there exist $m_1, m_2 \in \mathcal{M}$ such that $m = m_1 \bullet m_2$, $m_1 \in p$ and $m_2 \in \triangleright r$. Let

$$m = (l, s, U, \zeta) \qquad m_1 = (l_1, s, U, \zeta) \qquad m_2 = (l_2, s, U, \zeta)$$

Analogously to the read case, we only need to consider the case when $t \in \mathrm{dom}(U)$. From the definition of $\vdash_{\mathsf{fl}(t)}$ it follows that

$$(\mathrm{flush}(l_1, U(t)), s, U[t \mapsto \varepsilon], \zeta) \in \triangleright q.$$

By definition of action semantics we get $\lightning \notin [\![\mathit{fence}(t)]\!]$, since it never occurs for **fence**.

Now, take any $(h', U') \in [\![\mathit{fence}(t)]\!]$. By definition, this means that $h' = \mathrm{flush}(h, U(t))$ and $U' = U[t \mapsto \varepsilon]$. Take $m' = (\mathrm{flush}(l, U(t)), s, U', \zeta)$. Since flushing distributes over composition of local states, this state splits to $m'_1 = (\mathrm{flush}(l_1, U(t)), s, U', \zeta)$ and $m'_2 = (\mathrm{flush}(l_2, U(t)), s, U', \zeta)$. We already know $m'_1 \in \triangleright q$, and $m'_2 \in \triangleright r$ by virtue of stability of the latter. Thus, we only need to show that $(h', U') \in \lfloor m' \rfloor_A$. To this end, pick the state $l'_e = \mathrm{flush}(l_e, U(t))$ and the map $sr'(r) = \mathrm{flush}(sr(r), U(t))$, where l_e and sr are the witnesses of $(h, U) \in \lfloor m \rfloor_A$. This leaves us with showing $(sr'(r), U') \in \lfloor (s, \zeta) \rfloor_r$ for $r \in \mathrm{dom}\, s \cap A$. However, we know that $sr(r)\ R_{sb}\ sr'(r)$, and since $\lfloor (s, \zeta) \rfloor_r$ is defined as application of the action model, it's upwards-closed under interference. Thus, $sr_n(r) \in \lfloor (s, \zeta) \rfloor_r$, which ends the proof. □

Lemma 37. *For any $A \in \mathcal{P}(\Delta(\mathit{RId}))$, $p, q \in \mathit{Prop}_{\mathit{TSO}}$, $t \in \Delta(\mathit{TId})$, $o \in \Delta(\mathit{OId})$, $f \in \Delta(\mathit{FName})$, $v, v_o, v_n \in \Delta(\mathit{Val})$, $b \in \Delta(\mathbf{2})$,*

$$p \vdash_{\mathsf{fl}(t)} q \wedge p \vdash_{\mathrm{rd}(t)} o.f \mapsto v \wedge v \neq v_o \Rightarrow \mathit{cas}(t, o, f, v_o, v_n, b)\ \mathrm{sat}_A\ \{p\}\ \{q * b = \textbf{\textit{false}}\}$$

Proof. Let $r \in \mathit{Prop}_{\mathit{TSO}}$, $m \in \mathcal{M}$, $h \in \Delta(\mathit{Heap})$, and $U \in \Delta(\mathit{SPool})$ such that

$$m \in p * \triangleright r \qquad (h, U) \in \lfloor m \rfloor_A \qquad \underset{A}{\mathrm{stable}}(r).$$

Hence, there exists $m_1, m_2 \in \mathcal{M}$ such that $m = m_1 \bullet m_2$, $m_1 \in p$ and $m_2 \in \triangleright r$. As with read and write actions, if $t \notin \mathrm{dom}(U)$, the statement is vacuously true, since in that case $[\![\mathit{cas}(t, o, f, v_o, v_n, b)]\!] = \emptyset$. Thus, we know that $t \in \mathrm{dom}(U)$, and so, by definition of $\vdash_{\mathrm{rd}(t)}$, we get $\triangleright \mathrm{lookup}(o.f, U(t), m_1.l) = v$. As equalities on constant sets are upwards-closed, it follows that

$$\mathrm{lookup}(o.f, U(t), m_1.l) = v \vee \triangleright \bot$$

As the conclusion follows trivially in the $\triangleright\bot$ case (by Lemma 15), assume

$$\mathrm{lookup}(o.f, U(t), m_1.l) = v.$$

This means that $(o, f) \in \mathrm{dom}(m_1.l) \subseteq \mathrm{dom}(h)$, and so $\lightning \notin [\![\mathit{cas}(t, o, f, v_o, v_n, b)]\!]$.

Take any $(h', U') \in [\![\mathit{cas}(t, o, f, v_o, v_n, b)]\!](h, U)$. Since $v \neq v_o$, there is only one non-vacuous case, in which we learn that $h' = \mathrm{flush}(h, U(t))$, $U' = U[t \mapsto \varepsilon]$ and $b = \mathsf{false}$. Thus, we pick $m' = (\mathrm{flush}(m.l, U(t)), m.s, U', m.\zeta)$. This makes $(h', U') \in \lfloor m \rfloor_A$ hold by an argument analogous to the **fence** case. To show that

$$\triangleright(m \in q * \mathsf{false} = \mathsf{false} * r),$$

we pick the split of a local state into $l'_1 = \mathrm{flush}(m_1.l, U(t))$ and $l'_2 = \mathrm{flush}(m_2.l, U(t))$. The latter gives us $(l'_2, m.s, U', m.\zeta) \in \triangleright r$ by monotonicity of later and stability. On the other hand, $(l'_1, m.s, U', m.\zeta) \in \triangleright q$ by definition of $\vdash_{\mathsf{fl}(t)}$, while the final obligation is trivial. □

Lemma 38. *For any* $A \in \mathcal{P}(\Delta(RId))$*,* $p, q \in Prop_{TSO}$*,* $t \in \Delta(TId)$*,* $o \in \Delta(OId)$*,* $f \in \Delta(FName)$*,* $v_o, v_n \in \Delta(Val)$*,* $b \in \Delta(\mathbf{2})$*, if* $\mathrm{bstable}\, q$ *then*

$$p \vdash_{\mathrm{fl}(t)} \ulcorner o.f \mapsto v_o \urcorner * q \Rightarrow cas(t, o, f, v_o, v_n, b) \ \mathrm{sat}_A \ \{p\}\, \{\ulcorner o.f \mapsto v_n \urcorner * q * b = \textit{\textbf{true}}\}\,.$$

Proof. Let $r \in Prop_{TSO}$, $m \in \mathcal{M}$, $h \in \Delta(Heap)$ and $U \in \Delta(SPool)$ such that

$$m \in p(t) * \triangleright r \qquad (h, U) \in \lfloor m \rfloor_A \qquad \underset{A}{\mathrm{stable}}(r)$$

Hence, there exist $m_1, m_2 \in \mathcal{M}$ such that $m = m_1 \bullet m_2$, $m_1 \in p(t)$ and $m_2 \in \triangleright r$. Let

$$m = (l, s, U, \zeta) \qquad m_1 = (l_1, s, U, \zeta) \qquad m_2 = (l_2, s, U, \zeta)$$

As in the other cases, if $t \notin \mathrm{dom}(U)$, the statement holds vacuously. In the case when $t \in \mathrm{dom}(U)$, by definition of $\vdash_{\mathrm{fl}(t)}$ we get

$$(\mathrm{flush}(l_1, U(t)), s, U[t \mapsto \varepsilon], \zeta) \in \triangleright(\ulcorner o.f \mapsto v_o \urcorner * q),$$

from which we can deduce that $\triangleright\mathrm{lookup}(o.f, U(t), l_1) = v_o$, by the same argument as in the assignment case. Since equalities on constant sets are upwards-closed it follows that

$$\mathrm{lookup}(o.f, U(t), l_1) = v_o \vee \triangleright\bot$$

As the conclusion follows trivially by Lemma 15 in the $\triangleright\bot$ case, consider $\mathrm{lookup}(o.f, U(t), l_1) = v_o$. Since $(o, f) \in \mathrm{dom}(l_1) \subseteq \mathrm{dom}(h)$, we can conclude that $\lightning \notin [\![cas(t, o, f, v_o, v_n, b)]\!]$, which leaves us with the second conjunct.

Take any $(h', U') \in [\![cas(t, o, f, v_o, v_n, b)]\!](h, U)$. The only non-vacuous case is the successful compare-and-swap action, in which case $b = \textbf{true}$, $U' = U[t \mapsto \varepsilon]$, and $h' = \mathrm{flush}(h, U(t) \cdot (o, f, v_n))$. Pick $m' = (\mathrm{flush}(l, U(t) \cdot (o, f, v_n)), s, U[t \mapsto \varepsilon], \zeta)$. Since flushing distributes over $\bullet_{LState}$, we can easily show that

$$(\mathrm{flush}(l_2, U(t) \cdot (o, f, v_n)), s, U[t \mapsto \varepsilon], \zeta) \in \triangleright(r)$$

by using stability of r. By using the same argument as in the case of assignment we can also show

$$\triangleright(\mathrm{flush}(l_1, U(t) \cdot (o, f, v_n)), s, U[t \mapsto \varepsilon], \zeta) \in \ulcorner o.f \mapsto v_n \urcorner * q * \textbf{true} = \textbf{true}.$$

What remains is showing that $(h', U') \in \lfloor m' \rfloor_A$. However, note that since $o.f \in \mathrm{dom}(l_1)$, we know that $o.f \notin \mathrm{dom}(sr(r))$, where sr is the map obtained from $\lfloor m \rfloor_A$. Thus, it can be added to the store buffer by the interference relation, and the proof proceeds in an analogous way to the fence case. □

5.1 Read and Flush Judgments

Lemma 39 (Rd-Ax-Sound)**.** *For any $o \in \Delta(OId)$, $f \in \Delta(FName)$, $v \in \Delta(Val)$, $t \in \Delta(TId)$*

$$\triangleright\ulcorner o.f \mapsto v \textbf{ in } t\urcorner \vdash_{\mathrm{rd}(t)} o.f \mapsto v.$$

Proof. Take any $(l, s, U, \zeta) \in \triangleright\ulcorner o.f \mapsto v \textbf{ in } t\urcorner$ such that $t \in \mathrm{dom}(U)$. We have

$$\triangleright(\mathrm{flush}(l, U(t)), s, U[t \mapsto \varepsilon], \zeta) \in \ulcorner o.f \mapsto v\urcorner.$$

By definition, this means we have $\triangleright(l' \leq \mathrm{flush}(l, U(t)))$ such that $\triangleright(l', s, \zeta) \in o.f \mapsto v$, and so $\triangleright o.f \in \mathrm{dom}(l')$. From this, using properties of flush and lookup, we can conclude that $\triangleright\mathrm{lookup}(o.f, U(t), l) = v$, which ends the proof. $\square$

Lemma 40 (Rd-Cons)**.** *For any $p, q \in Prop_{TSO}$, $o \in \Delta(OId)$, $f \in \Delta(FName)$, $v \in \Delta(Val)$, $t \in \Delta(TId)$*

$$p \subseteq q \wedge q \vdash_{\mathrm{rd}(t)} o.f \mapsto v \Rightarrow p \vdash_{\mathrm{rd}(t)} o.f \mapsto v.$$

Proof. Take any $m \in p$ such that $t \in \mathrm{dom}(m.U)$. By the entailment assumption $m \in q$, so from the other assumption we get $\triangleright\mathrm{lookup}(o.f, m.U(t), m.l) = v$, which ends the proof. $\square$

Lemma 41 (Rd-$\mathcal{U}$-Eq)**.** *For any $p, q \in Prop_{TSO}$, $o \in \Delta(OId)$, $f \in \Delta(FName)$, $v \in \Delta(Val)$, $t \in \Delta(TId)$*

$$q \vdash_{\mathrm{rd}(t)} o.f \mapsto v \Rightarrow p\, \mathcal{U}_t\, q \vdash_{\mathrm{rd}(t)} o.f \mapsto v.$$

Proof. Take any $(l, s, U, \zeta) \in p\, \mathcal{U}_t\, q$ such that $t \in \mathrm{dom}(U)$. By definition of $\mathcal{U}_t$ we have α and β such that $U(t) = \alpha \cdot \beta$ and

$$(\mathrm{flush}(l, \alpha), s, U[t \mapsto \beta], \zeta) \in q.$$

This allows us to use the assumption, from which we can conclude that

$$\triangleright\mathrm{lookup}(o.f, \beta, \mathrm{flush}(l, \alpha)) = v,$$

and so, using the properties of lookup and flush, $\triangleright\mathrm{lookup}(o.f, \alpha \cdot \beta, l) = v$, which ends the proof. $\square$

Lemma 42 (Rd-$\mathcal{U}$-NEq)**.** *For any $p, q \in Prop_{TSO}$, $o \in \Delta(OId)$, $f \in \Delta(FName)$, $v \in \Delta(Val)$, $t, t' \in \Delta(TId)$, if $t \neq t'$ then*

$$p \vdash_{\mathrm{rd}(t)} o.f \mapsto v \Rightarrow p\, \mathcal{U}_{t'}\, q \vdash_{\mathrm{rd}(t)} o.f \mapsto v.$$

Proof. Take any $(l, s, U, \zeta) \in p \, \mathcal{U}_{t'} \, q$ such that $t \in \mathrm{dom}(U)$. By definition of $\mathcal{U}_t$ we get α and β such that $U(t') = \alpha \cdot \beta$ and $(l, s, U[t' \mapsto \alpha], \zeta) \in p$ (this time we fold the update into β). Now we can use the assumption, obtaining

$$\triangleright \mathrm{lookup}(o.f, U[t' \mapsto \alpha](t), l_2) = v,$$

from which it easily follows, since $t \neq t'$, that $\triangleright \mathrm{lookup}(o.f, U(t), l) = v$, which ends the proof. □

Lemma 43 (Fl-Ax-Sound)**.** *For any $p \in Prop_{SC}$, $t \in \Delta(TId)$,*

$$\triangleright \ulcorner p \textbf{ in } t \urcorner \vdash_{\mathrm{fl}(t)} \ulcorner p \urcorner.$$

Proof. Take any $(l, s, U, \zeta) \in \triangleright \ulcorner p \textbf{ in } t \urcorner$ such that $t \in \mathrm{dom}(U)$. By definition of interpretation in a thread, this gives us $(\mathrm{flush}(l, U(t)), s, U[t \mapsto \varepsilon], \zeta) \in \triangleright \ulcorner p \urcorner$, which is precisely the required obligation. □

Lemma 44 (Fl-*-Sound)**.** *For any $p, p', q, q' \in Prop_{TSO}$, $t \in \Delta(TId)$,*

$$p \vdash_{\mathrm{fl}(t)} p' \wedge q \vdash_{\mathrm{fl}(t)} q' \Rightarrow p * q \vdash_{\mathrm{fl}(t)} p' * q'.$$

Proof. Take any $(l, s, U, \zeta) \in p * q$ such that $t \in \mathrm{dom}(U)$. By definition of separating conjunction, this gives us l_1 and l_2 such that $l = l_1 \bullet l_2$, $(l_1, s, U, \zeta) \in p$ and $(l_2, s, U, \zeta) \in q$. From the assumptions, this gets us $(\mathrm{flush}(l_1, U(t)), s, U[t \mapsto \varepsilon], \zeta) \in \triangleright p'$ and $(\mathrm{flush}(l_2, U(t)), s, U[t \mapsto \varepsilon], \zeta) \in \triangleright q'$. Since flushing distributes over composition of local states, this gives us $(\mathrm{flush}(l, U(t)), s, U[t \mapsto \varepsilon], \zeta) \in \triangleright (p' * q')$, which ends the proof. □

Lemma 45 (Fl-Cons-Sound)**.** *For any $p, p', q, q' \in Prop_{TSO}$, $t \in \Delta(TId)$,*

$$p \subseteq p' \wedge p' \vdash_{\mathrm{fl}(t)} q' \wedge q' \subseteq q \Rightarrow p \vdash_{\mathrm{fl}(t)} q.$$

Proof. Take any $(l, s, U, \zeta) \in p$ such that $t \in \mathrm{dom}(U)$. By assumption, $(l, s, U, \zeta) \in p'$, which gives us $(\mathrm{flush}(l, U(t)), s, U[t \mapsto \varepsilon], \zeta) \in \triangleright q'$ by definition of $\vdash_{\mathrm{fl}(t)}$. By the final assumption, we get $(\mathrm{flush}(l, U(t)), s, U[t \mapsto \varepsilon], \zeta) \in \triangleright q$, which ends the proof. □

Lemma 46 (Fl-$\mathcal{U}$-Eq-Sound)**.** *For any $p, q, q' \in Prop_{TSO}$, $t \in \Delta(TId)$,*

$$q \vdash_{\mathrm{fl}(t)} q' \Rightarrow p \, \mathcal{U}_t \, q \vdash_{\mathrm{fl}(t)} q'.$$

Proof. Take any $(l, s, U, \zeta) \in p \, \mathcal{U}_t \, q$ such that $t \in \mathrm{dom}(U)$. From the definition of until, it follows that there exist α, β such that $U(t) = \alpha \cdot \beta$ and $(\mathrm{flush}(l, \alpha), s, U[t \mapsto \beta], \zeta) \in q$. We can use this state to instantiate our assumption, and obtain

$$(\mathrm{flush}(\mathrm{flush}(l, \alpha), \beta), s, U[t \mapsto \beta], \zeta) \in \triangleright q'.$$

However, this is equivalent to $(\mathrm{flush}(l, U(t)), s, U[t \mapsto \varepsilon], \zeta) \in \triangleright q'$, which ends the proof. □

Lemma 47 (Fl-$\mathcal{U}$-NEq-Sound)**.** *For any* $p, p' \in Prop_{TSO}$*,* $q \in Prop_{SC}$*,* $t, t' \in \Delta(TId)$*, if* $t \neq t'$ *then*

$$p \vdash_{\mathrm{fl}(t)} p' \Rightarrow p\, \mathcal{U}_{t'} \ulcorner q \urcorner \vdash_{\mathrm{fl}(t)} p'\, \mathcal{U}_{t'} \ulcorner q \urcorner.$$

Proof. Take any $(l, s, U, \zeta) \in p\ \mathcal{U}_t\ q$ such that $t \in \mathrm{dom}(U)$. By definition of until, there exist o, f, v, α and β, such that $U(t') = \alpha \cdot (o, f, v) \cdot \beta$, $(l, s, U[t' \mapsto \alpha], \zeta) \in p$ and $(\mathrm{flush}(l, \alpha \cdot (o, f, v)), s, U[t' \mapsto \beta], \zeta) \in \ulcorner q \urcorner$. We can use the first of these states to instantiate our assumption, and obtain:

$$(\mathrm{flush}(l, U(t)), s, U[t' \mapsto \alpha][t \mapsto \varepsilon], \zeta) \in \triangleright p'.$$

To show that the until holds, we pick the same splitting of $U(t)$. Thus, we are left with the following obligations:

$$(\mathrm{flush}(l, U(t)), s, U[t \mapsto \varepsilon][t' \mapsto \alpha], \zeta) \in \triangleright p'$$
$$(\mathrm{flush}(\mathrm{flush}(l, U(t)), \alpha \cdot (o, f, v)), s, U[t \mapsto \varepsilon][t' \mapsto \beta], \zeta) \in \ulcorner q \urcorner.$$

The first of these holds, since $t \neq t'$ and so the updates to the store buffers do not interfere. The second is more tricky. First, there exists a state $l' \leq l$ such that $(\mathrm{flush}(l', \alpha \cdot (o, f, v)), s, \zeta) \in q$ and $\mathrm{lfd}(l', U[t' \mapsto \beta])$. Thus, $\mathrm{flush}(l', U(t)) = l'$, and so we can pick the same smaller state. Also, since $U[t \mapsto \varepsilon][t' \mapsto \beta]$ contains a subset of the updates of $U[t' \mapsto \beta]$, and so the "locally flushed" condition holds. ◻

Theorem 1 (Soundness)**.** *If* $\Gamma \mid \Phi \vdash \mathsf{S}$ *then* $[\![\Gamma \mid \Phi \vdash \mathsf{S}]\!]$.

6 Verification of a spin-lock in the TSO logic

In this section we verify the spin-lock implementation using the TSO logic. We assume the reader has read the accompanying paper.

6.1 Specification

$$\begin{aligned}
&\exists \mathrm{isLock}, \mathrm{locked} : \mathsf{Prop_{SC}} \times \mathsf{Val} \to \mathsf{Prop_{SC}}.\\
&\quad \forall \mathsf{R} : \mathsf{Prop_{SC}}.\ \mathsf{stable}(\mathsf{R}) \ \Rightarrow\\
&\qquad\qquad [\overline{\mathsf{R}}]\ \mathsf{Lock()}\ [\mathsf{r}.\ \mathrm{isLock}(\mathsf{R}, \mathsf{r})]\\
&\qquad \wedge\ [\mathrm{isLock}(\mathsf{R}, \mathbf{this})]\ \mathsf{Lock.acquire()}\ [\mathrm{locked}(\mathsf{R}, \mathbf{this}) * \ulcorner \mathsf{R} \urcorner]\\
&\qquad \wedge\ [\mathrm{locked}(\mathsf{R}, \mathbf{this}) * \overline{\mathsf{R}}]\ \mathsf{Lock.release()}\ [\top]\\
&\qquad \wedge\ \mathsf{valid}(\forall \mathsf{x} : \mathsf{Val}.\ \mathrm{isLock}(\mathsf{R}, \mathsf{x}) \Leftrightarrow \mathrm{isLock}(\mathsf{R}, \mathsf{x}) * \mathrm{isLock}(\mathsf{R}, \mathsf{x}))\\
&\qquad \wedge\ \forall \mathsf{x} : \mathsf{Val}.\ \mathsf{stable}(\mathrm{isLock}(\mathsf{R}, \mathsf{x})) \wedge \mathsf{stable}(\mathrm{locked}(\mathsf{R}, \mathsf{x}))
\end{aligned}$$

Formally, the isLock and locked assertions are also embedded using $\overline{-}$, however as mentioned earlier, region assertions are independent of the local state and we thus elide the embeddings to make the proofs more readable.

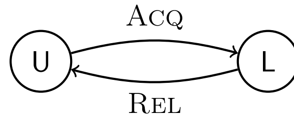
6.2 Predicate Definitions

$$\mathrm{isLock}(\mathsf{R},\mathsf{x}) = \exists \mathsf{n} : \mathsf{RId}.\ [\textsc{Acq}]^{\mathsf{n}}_{-} * \mathsf{region}(\{\mathsf{U},\mathsf{L}\}, \mathsf{T}_{lock}, I(\mathsf{x},\mathsf{R},\mathsf{n}), \mathsf{n})$$
$$\mathrm{locked}(\mathsf{R},\mathsf{x}) = \exists \mathsf{n} : \mathsf{RId}.\ [\textsc{Acq}]^{\mathsf{n}}_{-} * [\textsc{Rel}]^{\mathsf{n}}_{1} * \mathsf{region}(\{\mathsf{L}\}, \mathsf{T}_{lock}, I(\mathsf{x},\mathsf{R},\mathsf{n}), \mathsf{n})$$

where

$$I(\mathsf{x},\mathsf{R},\mathsf{n})(\mathsf{U}) = \exists \mathsf{t} : \mathsf{TId}.\ (\!\ulcorner \mathsf{x.locked} \mapsto \mathsf{true} \urcorner\, \mathcal{U}_{\mathsf{t}}\, \ulcorner \mathsf{x.locked} \mapsto \mathsf{false} * \mathsf{R} * [\textsc{Rel}]^{n}_{1} \urcorner\!)$$
$$I(\mathsf{x},\mathsf{R},\mathsf{n})(\mathsf{L}) = \ulcorner \mathsf{x.locked} \mapsto \mathsf{true} \urcorner$$

and T_{lock} refers to the following transition system



6.3 Proof outline

In this section we sketch proofs for each of the spin-lock methods in iCAP-TSO. Interestingly, the *structure* of the proof outlines differ in crucial ways from the corresponding proofs in iCAP (which assumes a strong memory model): In the case of acquire, even if the shared lock region is in the unlocked abstract state, it might still appear to be locked from the point of view of a client attempting to acquire the lock (if the lock was last released by a different thread, and release has not made its way to main memory yet), and the client will thus be unable to acquire the lock.

Constructor

```
class Lock {
  bool locked;

  Lock() =
[R̄ * ⌜this.locked ↦ null⌝]
  ⟨R̄ * ⌜this.locked ↦ null⌝⟩
    CAS(this.locked, false, null);
  ⟨⌜R * this.locked ↦ false⌝⟩
  ⟨region({U, L}, T_lock, I(this, R, n), n) * [ACQ]^n_−⟩
  ⟨isLock(R, this)⟩
[isLock(R, this)]
    this
[r. isLock(R, r)]
```

Acquire. The following is a proof sketch of the acquire method. As expected, most of the interesting reasoning concerns the atomic compare-and-swap expression that attempts to acquire the lock.

$\mathsf{acquire}() =$
$[\mathit{isLock}(\mathsf{R}, \mathbf{this})]$
$[\mathsf{region}(\{\mathsf{U},\mathsf{L}\}, \mathsf{T}_{lock}, I(\mathbf{this}, \mathsf{R}, \mathsf{n}), \mathsf{n}) * [\textsc{Acq}]^{\mathsf{n}}_{_}]$
$\quad \mathbf{let}\ \mathsf{y} = \mathsf{CAS}(\mathbf{this}.\mathsf{locked}, \mathbf{true}, \mathbf{false})\ \mathbf{in}$
$[(\mathsf{y} = \mathsf{true} * \mathsf{region}(\{\mathsf{L}\}, \mathsf{T}_{lock}, I(\mathbf{this}, \mathsf{R}, \mathsf{n}), \mathsf{n}) * [\textsc{Acq}]^{\mathsf{n}}_{_} * [\textsc{Rel}]^{\mathsf{n}}_{1} * \ulcorner \mathsf{R} \urcorner) \ \vee$
$\ (\mathsf{y} = \mathsf{false} * \mathsf{region}(\{\mathsf{U},\mathsf{L}\}, \mathsf{T}_{lock}, I(\mathbf{this}, \mathsf{R}, \mathsf{n}), \mathsf{n}) * [\textsc{Acq}]^{\mathsf{n}}_{_})]$
$\quad\quad \mathbf{if}\ \mathsf{y}\ \mathbf{then}$
$[\mathsf{region}(\{\mathsf{L}\}, \mathsf{T}_{lock}, I(\mathbf{this}, \mathsf{R}, \mathsf{n}), \mathsf{n}) * [\textsc{Acq}]^{n}_{_} * [\textsc{Rel}]^{\mathsf{n}}_{1} * \ulcorner \mathsf{R} \urcorner]$
$[\mathit{locked}(\mathsf{R}, \mathbf{this}) * \ulcorner \mathsf{R} \urcorner]$
$\quad\quad\quad ()$
$\quad\quad \mathbf{else}$
$[\mathsf{region}(\{\mathsf{U},\mathsf{L}\}, \mathsf{T}_{lock}, I(\mathbf{this}, \mathsf{R}, \mathsf{n}), \mathsf{n}) * [\textsc{Acq}]^{\mathsf{n}}_{_}]$
$[\mathit{isLock}(\mathsf{R}, \mathbf{this})]$
$\quad\quad\quad \mathsf{acquire}()$
$[\mathit{locked}(\mathsf{R}, \mathbf{this}) * \ulcorner \mathsf{R} \urcorner]$

To verify the atomic compare-and-swap we use the Atomic rule to open the shared lock region. Since the pre-condition asserts that the lock is either locked or unlocked ($\mathit{region}(\{\mathsf{U},\mathsf{L}\}, ...)$), we get two proof obligations, corresponding to each case:

$\langle [\textsc{Acq}]^{\mathsf{n}}_{_} * \triangleright I(\mathbf{this}, \mathsf{R}, \mathsf{n})(\mathsf{U}) \rangle$
$\quad \mathsf{CAS}(\mathbf{this}.\mathsf{locked}, \mathbf{true}, \mathbf{false})$
$\langle \mathsf{r}.\ \exists \mathsf{y} \in \{\mathsf{U},\mathsf{L}\}.\ [\textsc{Acq}]^{\mathsf{n}}_{_} * \triangleright I(\mathbf{this}, \mathsf{R}, \mathsf{n})(\mathsf{y}) * Q(\mathsf{y}, \mathsf{r}, \mathsf{n}) \rangle$

where

$$Q(\mathsf{y}, \mathsf{r}, \mathsf{n}) = (\mathsf{y} = \mathsf{L} * [\textsc{Rel}]^{\mathsf{n}}_{1} * \ulcorner \mathsf{R} \urcorner * \mathsf{r} = \mathsf{true}) \vee (\mathsf{y} = \mathsf{U} * \mathsf{r} = \mathsf{false})$$

and

$\langle [\textsc{Acq}]^{\mathsf{n}}_{_} * \triangleright I(\mathbf{this}, \mathsf{R}, \mathsf{n})(\mathsf{L}) \rangle$
$\quad \mathsf{CAS}(\mathbf{this}.\mathsf{locked}, \mathbf{true}, \mathbf{false})$
$\langle \mathsf{r}.\ [\textsc{Acq}]^{\mathsf{n}}_{_} * \triangleright I(\mathbf{this}, \mathsf{R}, \mathsf{n})(\mathsf{L}) * \mathsf{r} = \mathsf{false} \rangle$

We start with the (easy) second proof obligation:

$\langle [\textsc{Acq}]^{\mathsf{n}}_{_} * \triangleright \ulcorner \mathbf{this}.\mathsf{locked} \hookrightarrow \mathsf{true} \urcorner \rangle$
$\quad \langle \triangleright \ulcorner \mathbf{this}.\mathsf{locked} \hookrightarrow \mathsf{true} \urcorner \rangle$
$\quad \mathsf{CAS}(\mathbf{this}.\mathsf{locked}, \mathbf{true}, \mathbf{false})$
$\quad \langle \mathsf{r}.\ \triangleright \ulcorner \mathbf{this}.\mathsf{locked} \hookrightarrow \mathsf{true} \urcorner * \mathsf{r} = \mathsf{false} \rangle$
$\langle \mathsf{r}.\ [\textsc{Acq}]^{\mathsf{n}}_{_} * \triangleright \ulcorner \mathbf{this}.\mathsf{locked} \hookrightarrow \mathsf{true} \urcorner * \mathsf{r} = \mathsf{false} \rangle$

By rule A-Cas-False it thus suffices to prove that

$$\triangleright\ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{true}\urcorner \vdash^{\mathrm{m}}_{\mathrm{r}} \textbf{this}.\mathsf{locked} \mapsto \mathsf{true},$$

which follows easily by rules R-Self and R-Cons (to weaken $\ulcorner - \urcorner$ to $\overline{-}$).

To prove the first proof obligation, we first use Stab-$\mathcal{U}$ to do case analysis on whether or not the last release has made its way to main memory yet. Then we commute $\triangleright$ all the way into the pre-condition, before finally doing case analysis on whether (if the last release is still pending) the last release is in our store buffer or not.

$\langle[\mathrm{Acq}]^{\mathsf{n}}_{-} * \triangleright(\exists \mathsf{t} : \mathsf{TId}.\ (\!|\ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{true}\urcorner\, \mathcal{U}_{\mathsf{t}}\, \ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{false} * \mathsf{R} * [\mathrm{Rel}]^{\mathsf{n}}_{1}\urcorner|\!))\rangle$

$\quad\langle\triangleright(\exists \mathsf{t} : \mathsf{TId}.\ (\!|\ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{true}\urcorner\, \mathcal{U}_{\mathsf{t}}\, \ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{false} * \mathsf{R} * [\mathrm{Rel}]^{\mathsf{n}}_{1}\urcorner|\!))\rangle$

$\quad\langle\exists \mathsf{t} : \mathsf{TId}.\ \triangleright(\!|\ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{true}\urcorner\, \mathcal{U}_{\mathsf{t}}\, \ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{false} * \mathsf{R} * [\mathrm{Rel}]^{\mathsf{n}}_{1}\urcorner|\!))\rangle$

$\quad\quad\langle\triangleright(\!|\ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{true}\urcorner\, \mathcal{U}_{\mathsf{t}}\, \ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{false} * \mathsf{R} * [\mathrm{Rel}]^{\mathsf{n}}_{1}\urcorner|\!))\rangle$

$\quad\quad\langle\triangleright((\ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{true}\urcorner\, \mathcal{U}_{\mathsf{t}}\, (\!|\ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{false} * \mathsf{R} * [\mathrm{Rel}]^{\mathsf{n}}_{1}\urcorner|\!)\ \vee$
$\quad\quad\quad\quad\quad\quad (\!|\ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{false} * \mathsf{R} * [\mathrm{Rel}]^{\mathsf{n}}_{1}\urcorner|\!))\rangle$

$\quad\quad\langle\triangleright((\ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{true}\urcorner\, \mathcal{U}_{\mathsf{t}}\, \ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{false} * \mathsf{R} * [\mathrm{Rel}]^{\mathsf{n}}_{1}\urcorner)\ \vee$
$\quad\quad\quad\quad\quad\quad \ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{false} * \mathsf{R} * [\mathrm{Rel}]^{\mathsf{n}}_{1}\urcorner))\rangle$

$\quad\quad\langle((\triangleright\ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{true}\urcorner\, \mathcal{U}_{\mathsf{t}}\, \triangleright\ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{false}\urcorner * \triangleright\ulcorner\mathsf{R}\urcorner * \triangleright\ulcorner[\mathrm{Rel}]^{\mathsf{n}}_{1}\urcorner)\ \vee$
$\quad\quad\quad\quad\quad\quad \ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{false}\urcorner * \triangleright\ulcorner\mathsf{R}\urcorner * \triangleright\ulcorner[\mathrm{Rel}]^{\mathsf{n}}_{1}\urcorner))\rangle$

$\quad\quad\langle((\triangleright\ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{true}\urcorner\, \mathcal{U}\, \triangleright\ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{false}\urcorner * \triangleright\ulcorner\mathsf{R}\urcorner * \triangleright\ulcorner[\mathrm{Rel}]^{\mathsf{n}}_{1}\urcorner)\ \vee$
$\quad\quad\quad\quad (\triangleright\ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{true}\urcorner\, \mathcal{U}^{\mathrm{o}}\, \triangleright\ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{false}\urcorner * \triangleright\ulcorner\mathsf{R}\urcorner * \triangleright\ulcorner[\mathrm{Rel}]^{\mathsf{n}}_{1}\urcorner)\ \vee$
$\quad\quad\quad\quad\quad\quad \triangleright\ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{false}\urcorner * \triangleright\ulcorner\mathsf{R}\urcorner * \triangleright\ulcorner[\mathrm{Rel}]^{\mathsf{n}}_{1}\urcorner))\rangle$

$\quad\mathsf{CAS}(\textbf{this}.\mathsf{locked},\ \textbf{true},\ \textbf{false})$

$\quad\langle\mathsf{r}.\ \exists \mathsf{y} \in \{\mathsf{U}, \mathsf{L}\}.\ \triangleright I(\textbf{this}, \mathsf{R}, \mathsf{n})(\mathsf{y}) * Q(\mathsf{y}, \mathsf{r}, \mathsf{n})\rangle$

$\langle\mathsf{r}.\ [\mathrm{Acq}]^{\mathsf{n}}_{-} * \exists \mathsf{y} \in \{\mathsf{U}, \mathsf{L}\}.\ \triangleright I(\textbf{this}, \mathsf{R}, \mathsf{n})(\mathsf{y}) * Q(\mathsf{y}, \mathsf{r}, \mathsf{n})\rangle$

$\langle\mathsf{r}.\ \exists \mathsf{y} \in \{\mathsf{U}, \mathsf{L}\}.\ [\mathrm{Acq}]^{\mathsf{n}}_{-} * \triangleright I(\textbf{this}, \mathsf{R}, \mathsf{n})(\mathsf{y}) * Q(\mathsf{y}, \mathsf{r}, \mathsf{n})\rangle$

This leaves us with the following three proof obligations:

$\langle\triangleright\ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{true}\urcorner\, \mathcal{U}\, \triangleright\ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{false}\urcorner * \triangleright\ulcorner\mathsf{R}\urcorner * \triangleright\ulcorner[\mathrm{Rel}]^{\mathsf{n}}_{1}\urcorner\rangle$

$\quad\mathsf{CAS}(\textbf{this}.\mathsf{locked},\ \textbf{true},\ \textbf{false})$

$\langle\mathsf{r}.\ \exists \mathsf{y} \in \{\mathsf{U}, \mathsf{L}\}.\ \triangleright I(\textbf{this}, \mathsf{R}, \mathsf{n})(\mathsf{y}) * Q(\mathsf{y}, \mathsf{r}, \mathsf{n})\rangle$

and

$\langle\triangleright\ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{true}\urcorner\, \mathcal{U}^{\mathrm{o}}\, \triangleright\ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{false}\urcorner * \triangleright\ulcorner\mathsf{R}\urcorner * \triangleright\ulcorner[\mathrm{Rel}]^{\mathsf{n}}_{1}\urcorner)\rangle$

$\quad\mathsf{CAS}(\textbf{this}.\mathsf{locked},\ \textbf{true},\ \textbf{false})$

$\langle\mathsf{r}.\ \exists \mathsf{y} \in \{\mathsf{U}, \mathsf{L}\}.\ \triangleright I(\textbf{this}, \mathsf{R}, \mathsf{n})(\mathsf{y}) * Q(\mathsf{y}, \mathsf{r}, \mathsf{n})\rangle$

and

$\langle\triangleright\ulcorner\textbf{this}.\mathsf{locked} \mapsto \mathsf{false}\urcorner * \triangleright\ulcorner\mathsf{R}\urcorner * \triangleright\ulcorner[\mathrm{Rel}]^{n}_{1}\urcorner)\rangle$

$\quad\mathsf{CAS}(\textbf{this}.\mathsf{locked},\ \textbf{true},\ \textbf{false})$

$\langle\mathsf{r}.\ \exists \mathsf{y} \in \{\mathsf{U}, \mathsf{L}\}.\ \triangleright I(\textbf{this}, \mathsf{R}, \mathsf{n})(\mathsf{y}) * Q(\mathsf{y}, \mathsf{r}, \mathsf{n})\rangle$

corresponding to the three cases:

- that the last release is pending in our store buffer, or
- the last release is pending in some other thread's store buffer, or
- the last release has already made its way to main memory.

In the first and the last case, the lock is objectively *and* subjectively unlocked, and we can thus acquire the lock and transition to the locked state. However, in the second case, the lock is objectively unlocked, but subjectively locked, and we thus remain in the unlocked state. We thus strengthen the post-conditions of these three proof obligations as follows:

$\langle \triangleright\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{true}\urcorner\, \mathcal{U}\, \triangleright\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{false}\urcorner * \triangleright\ulcorner \mathsf{R}\urcorner * \triangleright\ulcorner [\textsc{Rel}]^{\mathsf{n}}_1\urcorner\rangle$
$\quad \mathsf{CAS}(\mathbf{this}.\mathsf{locked}, \mathbf{true}, \mathbf{false})$
$\langle \triangleright\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{true}\urcorner * [\textsc{Rel}]^{\mathsf{n}}_1 * \ulcorner \mathsf{R}\urcorner * \mathsf{r} = \mathsf{true}\rangle$
$\langle \mathsf{r}.\ \triangleright I(\mathbf{this}, \mathsf{R}, \mathsf{n})(\mathsf{L}) * Q(\mathsf{L}, \mathsf{r}, \mathsf{n})\rangle$

and

$\langle \triangleright\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{true}\urcorner\, \mathcal{U}^{\mathrm{o}}\, \triangleright\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{false}\urcorner * \triangleright\ulcorner \mathsf{R}\urcorner * \triangleright\ulcorner [\textsc{Rel}]^{\mathsf{n}}_1\urcorner)\rangle$
$\quad \mathsf{CAS}(\mathbf{this}.\mathsf{locked}, \mathbf{true}, \mathbf{false})$
$\langle \mathsf{r}.\ \triangleright\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{true}\urcorner\, \mathcal{U}^{\mathrm{o}}\, \triangleright\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{false}\urcorner * \triangleright\ulcorner \mathsf{R}\urcorner * \triangleright\ulcorner [\textsc{Rel}]^{\mathsf{n}}_1\urcorner) * \mathsf{r} = \mathsf{false}\rangle$
$\langle \mathsf{r}.\ \triangleright(\exists \mathsf{t} : \mathsf{Tld}.\ (\!(\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{true}\urcorner\, \mathcal{U}_{\mathsf{t}}\, \ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{false} * \mathsf{R} * [\textsc{Rel}]^{\mathsf{n}}_1\urcorner)\!)) * \mathsf{r} = \mathsf{false}\rangle$
$\langle \mathsf{r}.\ \triangleright I(\mathbf{this}, \mathsf{R}, \mathsf{n})(\mathsf{U}) * Q(\mathsf{U}, \mathsf{r}, \mathsf{n})\rangle$

and

$\langle \triangleright\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{false}\urcorner * \triangleright\ulcorner \mathsf{R}\urcorner * \triangleright\ulcorner [\textsc{Rel}]^{\mathsf{n}}_1\urcorner)\rangle$
$\quad \mathsf{CAS}(\mathbf{this}.\mathsf{locked}, \mathbf{true}, \mathbf{false})$
$\langle \mathsf{r}.\ \triangleright\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{true}\urcorner * [\textsc{Rel}]^{\mathsf{n}}_1 * \ulcorner \mathsf{R}\urcorner * \mathsf{r} = \mathsf{true}\rangle$
$\langle \mathsf{r}.\ \triangleright I(\mathbf{this}, \mathsf{R}, \mathsf{n})(\mathsf{L}) * Q(\mathsf{L}, \mathsf{r}, \mathsf{n})\rangle$

By rules A-CAS-False and A-CAS-True, these three proof obligations reduce to the following three read/write proof obligations:

$$\triangleright\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{true}\urcorner\, \mathcal{U}\, \triangleright\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{false}\urcorner * \triangleright\ulcorner \mathsf{R}\urcorner * \triangleright\ulcorner [\textsc{Rel}]^{\mathsf{n}}_1\urcorner$$
$$\vdash_{\mathrm{w}} \mathbf{this}.\mathsf{locked} \mapsto \mathsf{false}, [\textsc{Rel}]^{\mathsf{n}}_1 * \mathsf{R}$$

and

$$\triangleright\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{true}\urcorner\, \mathcal{U}^{\mathrm{o}}\, \triangleright\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{false}\urcorner * \triangleright\ulcorner \mathsf{R}\urcorner * \triangleright\ulcorner [\textsc{Rel}]^{\mathsf{n}}_1\urcorner \vdash^{\mathrm{m}}_{\mathrm{r}} \mathbf{this}.\mathsf{locked} \mapsto \mathsf{true}$$

and

$$\triangleright\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{false}\urcorner * \triangleright\ulcorner \mathsf{R}\urcorner * \triangleright\ulcorner [\textsc{Rel}]^{\mathsf{n}}_1\urcorner \vdash_{\mathrm{w}} \mathbf{this}.\mathsf{locked} \mapsto \mathsf{false}, [\textsc{Rel}]^{\mathsf{n}}_1 * \mathsf{R}$$

In the first and last case we can use rule W-Cons to weaken $\ulcorner - \urcorner$ to $\overline{=}$ and W-$*$ to frame off R and $[\textsc{Rel}]$ and remove the laters in front of these. It thus suffices to prove:

$$\triangleright\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{true}\urcorner\, \mathcal{U}\, \triangleright\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{false}\urcorner \vdash_{\mathrm{w}} \mathbf{this}.\mathsf{locked} \mapsto \mathsf{false}, \top$$
$$\triangleright\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{true}\urcorner\, \mathcal{U}^{\mathrm{o}}\, \triangleright\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{false}\urcorner \vdash^{\mathrm{m}}_{\mathrm{r}} \mathbf{this}.\mathsf{locked} \mapsto \mathsf{true}$$
$$\triangleright\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{false}\urcorner \vdash_{\mathrm{w}} \mathbf{this}.\mathsf{locked} \mapsto \mathsf{false}, \top$$

The last proof obligation follows easily from W-Ax and W-Cons. By rules W-$\mathcal{U}$ and R-$\mathcal{U}^{o}$, the first two obligations reduce to:

$$\triangleright\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{false} \urcorner \vdash_{w} \mathbf{this}.\mathsf{locked} \mapsto \mathsf{false}, \top$$
$$\triangleright\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{true} \urcorner \vdash_{r}^{o} \mathbf{this}.\mathsf{locked} \mapsto \mathsf{true}$$

These follow easily using rules W-Ax, W-Cons and R-Other, R-Cons, respectively.

Release. Below we sketch the proof outline of release:

release() =
$[\mathrm{locked}(\mathsf{R}, \mathbf{this}) * \overline{\mathsf{R}}]$
$[\mathsf{region}(\{\mathsf{L}\}, \mathsf{T}_{lock}, I(\mathbf{this}, \mathsf{R}, \mathsf{n}), \mathsf{n}) * [\mathrm{Acq}]^{\mathsf{n}}_{_} * [\mathrm{Rel}]^{\mathsf{n}}_{1} * \overline{\mathsf{R}}]$
  $[\triangleright\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{true} \urcorner * [\mathrm{Acq}]^{\mathsf{n}}_{_} * [\mathrm{Rel}]^{\mathsf{n}}_{1} * \overline{\mathsf{R}}]$
    $[\triangleright\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{true} \urcorner * [\mathrm{Rel}]^{\mathsf{n}}_{1} * \overline{\mathsf{R}}]$
  this.locked := **false**
    $[(\triangleright\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{true} \urcorner * [\mathrm{Rel}]^{\mathsf{n}}_{1} * \overline{\mathsf{R}})\ \mathcal{U}\ ([\mathrm{Rel}]^{\mathsf{n}}_{1} * \ulcorner \mathsf{R} \urcorner * \ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{false} \urcorner)]$
    $[\triangleright(\exists \mathsf{t} : \mathsf{TId}.\ (\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{true} \urcorner\ \mathcal{U}_{\mathsf{t}}\ \ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{false} * \mathsf{R} * [\mathrm{Rel}]^{\mathsf{n}}_{1} \urcorner))]$
  $[\triangleright(\exists \mathsf{t} : \mathsf{TId}.\ (\ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{true} \urcorner\ \mathcal{U}_{\mathsf{t}}\ \ulcorner \mathbf{this}.\mathsf{locked} \mapsto \mathsf{false} * \mathsf{R} * [\mathrm{Rel}]^{\mathsf{n}}_{1} \urcorner)) * [\mathrm{Acq}]^{\mathsf{n}}_{_}]$
  $[\triangleright I(\mathbf{this}, \mathsf{R}, \mathsf{n})(\mathsf{U}) * [\mathrm{Acq}]^{\mathsf{n}}_{_}]$
$[\mathsf{region}(\{\mathsf{U}, \mathsf{L}\}, \mathsf{T}_{lock}, I(\mathbf{this}, \mathsf{R}, \mathsf{n}), \mathsf{n}) * [\mathrm{Acq}]^{\mathsf{n}}_{_}]$
$[\mathrm{isLock}(\mathsf{R}, \mathbf{this})]$

The innermost proof obligation follows by A-Write.

7 Logical Atomicity: Treiber's Stack

In this section we illustrate that the TSO logic is sufficiently expressive to express logical atomicity using the specification pattern we introduced in [8]. In particular, we illustrate the verification of Treiber's stack against a TSO specification which expresses that the effects of the push and pop method appear to take affect atomically. Furthermore, we illustrate how this specification allows us to *derive* more standard specifications in the SC logic.

7.1 Specification

Following iCAP [7], we express logical atomicity (there referred to as granularity abstraction) by reasoning about the abstract state of the stack through a phantom field and parameterizing the specification of push and pop with view-shifts to update the clients knowledge about the abstract state, when it changes atomically. Crucially, Treiber's stack ensures that when the abstract state changes, the store buffer of the thread that causes the change is also flushed; hence, if $\overline{\mathsf{P}(\mathsf{x}, \mathsf{y})}$ holds before the execution of push, $\ulcorner \mathsf{P}(\mathsf{x}, \mathsf{y}) \urcorner$ holds when the abstract effects of push appear to take affect. We will exploit

this to derive a more standard SC specification in Section 7.4. The full specification is given below.

$$
\begin{array}{l}
\exists stack : \mathsf{RId} \times \mathsf{Val} \to \mathsf{Prop_{SC}}.\\
\quad [\mathsf{emp}]\\
\qquad \mathsf{new\ Stack}(-)\\
\quad \left[\mathsf{r}.\ \exists \mathsf{n} : \mathsf{RId}.\ stack(\mathsf{n}, \mathsf{r}) * \mathsf{r_{cont}} \overset{1/2}{\longmapsto} \varepsilon\right]
\end{array}
$$

$$
\begin{array}{l}
\forall \mathsf{P}, \mathsf{Q} : \mathsf{Val} \times \mathsf{Val} \to \mathsf{Prop_{SC}}.\ \forall \mathsf{n} : \mathsf{RId}.\\
\quad (\forall \mathsf{x}, \mathsf{y} : \mathsf{Val}.\ \mathsf{stable}(\mathsf{P}(\mathsf{x}, \mathsf{y})) \wedge \mathsf{stable}(\mathsf{Q}(\mathsf{x}, \mathsf{y}))\ \wedge\\
\quad (\forall \alpha : \mathrm{seq}\ \mathsf{Val}.\ \forall \mathsf{x}, \mathsf{y} : \mathsf{Val}.\ \triangleright \mathsf{x_{cont}} \overset{1/2}{\longmapsto} \alpha * \ulcorner \mathsf{P}(\mathsf{x}, \mathsf{y}) \urcorner \sqsubseteq^{\mathsf{RId} \setminus \{\mathsf{n}\}} \triangleright(\mathsf{x_{cont}} \overset{1/2}{\longmapsto} (\mathsf{y} :: \alpha) * \ulcorner \mathsf{Q}(\mathsf{x}, \mathsf{y}) \urcorner))\ \Rightarrow\\
\qquad \left[stack(\mathsf{n}, \mathsf{x}) * \overline{\mathsf{P}(\mathsf{x}, \mathsf{y})}\right]\\
\qquad\quad \mathsf{x.push}(\mathsf{y})\\
\qquad [stack(\mathsf{n}, \mathsf{x}) * \ulcorner \mathsf{Q}(\mathsf{x}, \mathsf{y}) \urcorner]
\end{array}
$$

$$
\begin{array}{l}
\forall \mathsf{P} : \mathsf{Val} \to \mathsf{Prop_{SC}}.\ \forall \mathsf{Q} : \mathsf{Val} \times \mathsf{Val} \to \mathsf{Prop_{SC}}.\ \forall \mathsf{n} : \mathsf{RId}.\\
\quad (\forall \mathsf{x}, \mathsf{y} : \mathsf{Val}.\ \mathsf{stable}(\mathsf{P}(\mathsf{x})) \wedge \mathsf{stable}(\mathsf{Q}(\mathsf{x}, \mathsf{y}))\ \wedge\\
\quad (\forall \alpha : \mathrm{seq}\ \mathsf{Val}.\ \forall \mathsf{x}, \mathsf{y} : \mathsf{Val}.\ \triangleright \mathsf{x_{cont}} \overset{1/2}{\longmapsto} (\mathsf{y} :: \alpha) * \ulcorner \mathsf{P}(\mathsf{x}) \urcorner \sqsubseteq^{\mathsf{RId} \setminus \{\mathsf{n}\}} \triangleright(\mathsf{x_{cont}} \overset{1/2}{\longmapsto} \alpha * \ulcorner \mathsf{Q}(\mathsf{x}, \mathsf{y}) \urcorner))\ \wedge\\
\quad (\forall \mathsf{x} : \mathsf{Val}.\ \triangleright \mathsf{x_{cont}} \overset{1/2}{\longmapsto} \varepsilon * \overline{\mathsf{P}(\mathsf{x})} \sqsubseteq^{\mathsf{RId} \setminus \{\mathsf{n}\}} \triangleright(\mathsf{x_{cont}} \overset{1/2}{\longmapsto} \varepsilon * \ulcorner \mathsf{Q}(\mathsf{x}, \mathbf{null}) \urcorner))\ \Rightarrow\\
\qquad \left[stack(\mathsf{n}, \mathsf{x}) * \overline{\mathsf{P}(\mathsf{x})}\right]\\
\qquad\quad \mathsf{x.pop}(-)\\
\qquad [\mathsf{r}.\ stack(\mathsf{n}, \mathsf{x}) * \ulcorner \mathsf{Q}(\mathsf{x}, \mathsf{r}) \urcorner]
\end{array}
$$

7.2 Predicate definitions

$$stack(\mathsf{x}) = \exists \mathsf{r} : \mathsf{RId}.\ \mathsf{region}(\{s\}, T, \mathsf{I}(\mathsf{x}), \mathsf{r})$$

where T is a labelled transition system with a single state s and I is defined as follows

$$\mathsf{I}(\mathsf{x})(s) = \exists \mathsf{y} : \mathsf{Val}.\ \exists \alpha : \mathrm{seq}\ \mathsf{Val}.\ \ulcorner \mathsf{x.head} \mapsto \mathsf{y} * lst_r(\mathsf{y}, \alpha) * \triangleright \mathsf{x_{cont}} \overset{1/2}{\longmapsto} \alpha \urcorner$$

and $lst_r : \mathsf{Val} \times \mathrm{seq}\ \mathsf{Val} \to \mathsf{Prop_{SC}}$ is defined as follows by induction on its second argument:

$$
\begin{aligned}
lst_r(\mathsf{x}, \varepsilon) &= \mathsf{x} =_{\mathsf{Val}} \mathbf{null}\\
lst_r(\mathsf{x}, \mathsf{y} :: \alpha) &= \exists \mathsf{z} : \mathsf{Val}.\ \mathsf{x.next} \mapsto \mathsf{z} * \mathsf{x.val} \mapsto \mathsf{y} * lst_r(\mathsf{z}, \alpha)
\end{aligned}
$$

7.3 Proof outline

class Node {
  Object val;
  Node next;

  Node(Object x) =
$[\ulcorner \mathbf{this}.\mathsf{val} \mapsto _ * \mathbf{this}.\mathsf{next} \mapsto _ \urcorner]$
    this.val = x;
$[\overline{\mathbf{this}.\mathsf{val} \mapsto \mathsf{x} * \mathbf{this}.\mathsf{next} \mapsto _}]$
    this
}

class Stack {
  Node head;

  Stack() =
$[\ulcorner \mathbf{this}.\mathsf{head} \mapsto _ \urcorner * \mathbf{this}_{\mathsf{cont}} \mapsto \varepsilon]$
    this.head = null;
$[\overline{\mathbf{this}.\mathsf{head} \mapsto \mathbf{null}} * \mathbf{this}_{\mathsf{cont}} \mapsto \varepsilon]$
    fence;
$[\ulcorner \mathbf{this}.\mathsf{head} \mapsto \mathbf{null} * lst_r(\mathbf{null}, \varepsilon) \urcorner * \mathbf{this}_{\mathsf{cont}} \mapsto \varepsilon]$
$[stack(\mathbf{this}) * \mathbf{this}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \varepsilon]$
    this
  }

  unit push(Object x) =
$[stack(\mathbf{this}) * \overline{\mathsf{P}(\mathbf{this}, \mathsf{x})}]$
    let nHead = new Node(x) **in**
$[stack(\mathbf{this}) * \overline{\mathsf{P}(\mathbf{this}, \mathsf{x}) * \mathsf{nHead.val} \mapsto \mathsf{x} * \mathsf{nHead.next} \mapsto _}]$
      push'(nHead)
$[stack(\mathbf{this}) * \ulcorner \mathsf{Q}(\mathbf{this}, \mathsf{x}) \urcorner]$

  unit push'(Node nHead) =
$[stack(\mathbf{this}) * \overline{\mathsf{P}(\mathbf{this}, \mathsf{x}) * \mathsf{nHead.val} \mapsto \mathsf{x} * \mathsf{nHead.next} \mapsto _}]$
    let oHead = **this**.head **in**
$[stack(\mathbf{this}) * \overline{\mathsf{P}(\mathbf{this}, \mathsf{x}) * \mathsf{nHead.val} \mapsto \mathsf{x} * \mathsf{nHead.next} \mapsto _}]$
      nHead.next := oHead;
$[stack(\mathbf{this}) * \overline{\mathsf{P}(\mathbf{this}, \mathsf{x}) * \mathsf{nHead.val} \mapsto \mathsf{x} * \mathsf{nHead.next} \mapsto \mathsf{oHead}}]$
      let t = CAS(**this**.head, nHead, oHead) **in**
$[stack(\mathbf{this}) *$
    $((t = \mathbf{true} * \triangleright \ulcorner \mathsf{Q}(\mathbf{this}, \mathsf{x}) \urcorner) \quad \vee$

$(t = \mathsf{false} * \overline{\mathsf{P}(\mathbf{this}, \mathsf{x}) * \mathsf{nHead.val} \mapsto \mathsf{x} * \mathsf{nHead.next} \mapsto \mathsf{oHead}}))]$
if t **then** () **else** push'(nHead)
$[stack(\mathbf{this}) * \ulcorner \mathsf{Q}(\mathbf{this}, \mathsf{x}) \urcorner]$

As always, the main difficulties in verifying the push′ method, is the verification of the two atomic expressions that interact with the shared state. The first of these, **this**.head simply reads the value of a shared field, but since this is simply an optimistic read, whose value will be checked later by the CAS, we do not need to retain any information about the value read. To verify this atomic expression it thus suffices to prove that we have permission to read it. This follows by opening the region and framing:

$[stack(\mathbf{this}) * \overline{\mathsf{P}(\mathbf{this}, \mathsf{x}) * \mathsf{nHead.val} \mapsto \mathsf{x} * \mathsf{nHead.next} \mapsto _}]$
$\langle stack(\mathbf{this}) \rangle$
$\langle \mathsf{region}(\{s\}, T, I(\mathbf{this}), \mathsf{r}) \rangle$
$\langle \triangleright I(\mathbf{this})(s) \rangle$
$\langle \triangleright \ulcorner \mathbf{this}.\mathsf{head} \mapsto \mathsf{y} \urcorner \rangle$
this.head
$\langle \mathsf{r}.\ \ulcorner \mathbf{this}.\mathsf{head} \mapsto \mathsf{y} \urcorner * \mathsf{r} = \mathsf{y} \rangle$
$\langle \mathsf{r}.\ I(\mathbf{this})(s) \rangle$
$\langle \mathsf{r}.\ \mathsf{region}(\{s\}, T, I(\mathbf{this}), \mathsf{r}) \rangle$
$\langle \mathsf{r}.\ stack(\mathbf{this}) \rangle$
$[\mathsf{r}.\ stack(\mathbf{this}) * \overline{\mathsf{P}(\mathbf{this}, \mathsf{x}) * \mathsf{nHead.val} \mapsto \mathsf{x} * \mathsf{nHead.next} \mapsto _}]$

The second expression is the CAS that attempts to swing the head pointer.

$[stack(\mathbf{this}) * \overline{\mathsf{R}}]$
$\langle stack(\mathbf{this}) * \overline{\mathsf{R}} \rangle$
$\langle \mathsf{region}(\{s\}, T, I(\mathbf{this}), \mathsf{r}) * \overline{\mathsf{R}} \rangle$
$\langle \triangleright I(\mathbf{this})(s) * \overline{\mathsf{R}} \rangle$
$\langle \triangleright (\ulcorner \mathbf{this}.\mathsf{head} \mapsto \mathsf{y} * lst_r(\mathsf{y}, \alpha) * \triangleright \mathbf{this}_{\mathsf{cont}} \xmapsto{1/2} \alpha \urcorner) * \overline{\mathsf{R}} \rangle$
CAS(**this**.head, nHead, oHead) **in**
$\langle \mathsf{r}.\ (\mathsf{r} = \mathbf{true} * \mathsf{y} = \mathsf{oHead} * \ulcorner \mathbf{this}.\mathsf{head} \mapsto \mathsf{nHead} * lst_r(\mathsf{y}, \alpha) * \triangleright \mathbf{this}_{\mathsf{cont}} \xmapsto{1/2} \alpha * \mathsf{R} \urcorner)$
$(\mathsf{r} = \mathbf{false} * \mathsf{y} \neq \mathsf{oHead} * \ulcorner \mathbf{this}.\mathsf{head} \mapsto \mathsf{nHead} * lst_r(\mathsf{y}, \alpha) * \triangleright \mathbf{this}_{\mathsf{cont}} \xmapsto{1/2} \alpha \urcorner * \overline{\mathsf{R}}) \rangle$
$\langle \mathsf{r}.\ (\mathsf{r} = \mathbf{true} * \ulcorner \mathbf{this}.\mathsf{head} \mapsto \mathsf{nHead} * lst_r(\mathsf{nHead}, \mathsf{x} :: \alpha) * \triangleright \mathbf{this}_{\mathsf{cont}} \xmapsto{1/2} \alpha * \mathsf{P}(\mathbf{this}, \mathsf{x}) \urcorner)$
$(\mathsf{r} = \mathbf{false} * I(\mathbf{this})(s) * \overline{\mathsf{R}}) \rangle$
$\langle \mathsf{r}.\ (\mathsf{r} = \mathbf{true} * \ulcorner \mathbf{this}.\mathsf{head} \mapsto \mathsf{nHead} * lst_r(\mathsf{nHead}, \mathsf{x} :: \alpha) * \triangleright \mathbf{this}_{\mathsf{cont}} \xmapsto{1/2} (\mathsf{x} :: \alpha) * \triangleright \mathsf{Q}(\mathbf{this}, \mathsf{x}) \urcorner)$
$(\mathsf{r} = \mathbf{false} * I(\mathbf{this})(s) * \overline{\mathsf{R}}) \rangle$
$\langle \mathsf{r}.\ I(\mathbf{this})(s) * ((\mathsf{r} = \mathbf{true} * \triangleright \ulcorner \mathsf{Q}(\mathbf{this}, \mathsf{x}) \urcorner) \vee (\mathsf{r} = \mathbf{false} * \overline{\mathsf{R}})) \rangle$
$\langle \mathsf{r}.\ \mathsf{region}(\{s\}, T, I(\mathbf{this}), \mathsf{r}) * ((\mathsf{r} = \mathbf{true} * \triangleright \ulcorner \mathsf{Q}(\mathbf{this}, \mathsf{x}) \urcorner) \vee (\mathsf{r} = \mathbf{false} * \overline{\mathsf{R}})) \rangle$
$\langle \mathsf{r}.\ stack(\mathbf{this}) * ((\mathsf{r} = \mathbf{true} * \triangleright \ulcorner \mathsf{Q}(\mathbf{this}, \mathsf{x}) \urcorner) \vee (\mathsf{r} = \mathbf{false} * \overline{\mathsf{R}})) \rangle$
$[\mathsf{r}.\ stack(\mathbf{this}) * ((\mathsf{r} = \mathbf{true} * \triangleright \ulcorner \mathsf{Q}(\mathbf{this}, \mathsf{x}) \urcorner) \vee (\mathsf{r} = \mathbf{false} * \overline{\mathsf{R}}))]$

where R is shorthand for $\mathsf{P}(\mathbf{this}, \mathsf{x}) * \mathsf{nHead.val} \mapsto \mathsf{x} * \mathsf{nHead.next} \mapsto \mathsf{oHead}$.

The proof of pop also proceeds exactly like the corresponding proof in iCAP. First we read the head field and do case-analysis on whether the value we read is **null** or not; if it is, we apply the empty stack view-shift; otherwise, we keep a copy of the read-only lst_r assertion from the stack region, to retain the knowledge that head refers to a linked list. This allows us to follow the next field, in case `oHead` is non-null.

$$
\begin{array}{l}
\quad \mathsf{Object\ pop() =} \\
[\mathit{stack}(\mathbf{this}) * \overline{\mathsf{P}(\mathbf{this})}] \\
\qquad \mathbf{let}\ \mathsf{oHead} = \mathbf{this}.\mathsf{head}\ \mathbf{in} \\
[\mathit{stack}(\mathbf{this}) * ((\mathsf{oHead} = \mathbf{null} * \triangleright\ulcorner\mathsf{Q}(\mathbf{this}, \mathbf{null})\urcorner) \vee (\mathsf{oHead} \neq \mathbf{null} * \ulcorner lst_r(\mathsf{oHead}, _)\urcorner * \overline{\mathsf{P}(\mathbf{this})}))] \\
\qquad\quad \mathbf{if}\ \mathsf{oHead} = \mathsf{null}\ \mathbf{then} \\
[\mathit{stack}(\mathbf{this}) * \triangleright\ulcorner\mathsf{Q}(\mathbf{this}, \mathbf{null})\urcorner] \\
\qquad\qquad \mathsf{null} \\
[\mathsf{r}.\mathit{stack}(\mathbf{this}) * \ulcorner\mathsf{Q}(\mathbf{this}, \mathsf{r})\urcorner] \\
\qquad\quad \mathbf{else} \\
[\mathit{stack}(\mathbf{this}) * \mathsf{oHead} \neq \mathbf{null} * \ulcorner lst_r(\mathsf{oHead}, _)\urcorner * \overline{\mathsf{P}(\mathbf{this})}] \\
\qquad\qquad \mathbf{let}\ \mathsf{nHead} = \mathsf{oHead.next}\ \mathbf{in} \\
[\mathit{stack}(\mathbf{this}) * \ulcorner\mathsf{oHead.val} \mapsto \mathsf{y} * \mathsf{oHead.next} \mapsto \mathsf{nHead} * lst_r(\mathsf{nHead}, _)\urcorner * \overline{\mathsf{P}(\mathbf{this})}] \\
\qquad\qquad \mathbf{let}\ \mathsf{t} = \mathsf{CAS}(\mathbf{this}.\mathsf{head}, \mathsf{nHead}, \mathsf{oHead})\ \mathbf{in} \\
[\mathit{stack}(\mathbf{this}) * ((\mathsf{t} = \mathbf{true} * \ulcorner\triangleright\mathsf{Q}(\mathbf{this}, \mathsf{y}) * \mathsf{oHead.val} \mapsto \mathsf{y}\urcorner) \vee (\mathsf{t} = \mathbf{false} * \overline{\mathsf{P}(\mathbf{this})}))] \\
\qquad\qquad\quad \mathbf{if}\ \mathsf{t}\ \mathbf{then}\ \mathsf{oHead.val}\ \mathbf{else}\ \mathsf{pop()} \\
[\mathsf{r}.\ \mathit{stack}(\mathbf{this}) * \ulcorner\mathsf{Q}(\mathbf{this}, \mathsf{r})\urcorner] \\
\}
\end{array}
$$

Again we are left with two proof obligations, one for each of the atomic expressions that access the shared state. As before, the first of these, **this**.head, simply reads the value of the shared head field; however, as explained above, depending on whether we read **null**, we either have to apply a view-shift or retain the lst_r assertion:

$$
\begin{array}{l}
[\mathit{stack}(\mathbf{this}) * \overline{\mathsf{P}(\mathbf{this})}] \\
\langle\mathit{stack}(\mathbf{this}) * \overline{\mathsf{P}(\mathbf{this})}\rangle \\
\langle\mathsf{region}(\{s\}, T, I(\mathbf{this}), \mathsf{r}) * \overline{\mathsf{P}(\mathbf{this})}\rangle \\
\langle\triangleright I(\mathbf{this})(s) * \overline{\mathsf{P}(\mathbf{this})}\rangle \\
\langle\triangleright(\ulcorner\mathbf{this}.\mathsf{head} \mapsto \mathsf{y} * lst_r(\mathsf{y}, \alpha) * \triangleright\mathbf{this}_{\mathsf{cont}} \xmapsto{1/2} \alpha\urcorner) * \overline{\mathsf{P}(\mathbf{this})}\rangle \\
\quad \mathbf{this}.\mathsf{head} \\
\langle\mathsf{r}.\ \ulcorner\mathbf{this}.\mathsf{head} \mapsto \mathsf{y} * lst_r(\mathsf{y}, \alpha) * \triangleright\mathbf{this}_{\mathsf{cont}} \xmapsto{1/2} \alpha\urcorner * \mathsf{r} = \mathsf{y} * \overline{\mathsf{P}(\mathbf{this})}\rangle \\
\langle\mathsf{r}.\ (\mathsf{r} = \mathbf{null} * \ulcorner\mathbf{this}.\mathsf{head} \mapsto \mathbf{null} * lst_r(\mathbf{null}, \alpha) * \triangleright\mathbf{this}_{\mathsf{cont}} \xmapsto{1/2} \alpha\urcorner * \overline{\mathsf{P}(\mathbf{this})}) \\
\qquad (\mathsf{r} \neq \mathbf{null} * \ulcorner\mathbf{this}.\mathsf{head} \mapsto \mathsf{r} * lst_r(\mathsf{r}, \alpha) * \triangleright\mathbf{this}_{\mathsf{cont}} \xmapsto{1/2} \alpha\urcorner * \overline{\mathsf{P}(\mathbf{this})})\rangle \\
\langle\mathsf{r}.\ (\mathsf{r} = \mathbf{null} * \ulcorner\mathbf{this}.\mathsf{head} \mapsto \mathbf{null} * lst_r(\mathbf{null}, \varepsilon) * \triangleright\mathbf{this}_{\mathsf{cont}} \xmapsto{1/2} \varepsilon\urcorner * \triangleright\ulcorner\mathsf{Q}(\mathbf{this}, \mathbf{null})\urcorner) \\
\qquad (\mathsf{r} \neq \mathbf{null} * \ulcorner\mathbf{this}.\mathsf{head} \mapsto \mathsf{r} * lst_r(\mathsf{r}, \alpha) * lst_r(\mathsf{r}, \alpha) * \triangleright\mathbf{this}_{\mathsf{cont}} \xmapsto{1/2} \alpha\urcorner * \overline{\mathsf{P}(\mathbf{this})})\rangle \\
\langle\mathsf{r}.\ I(\mathbf{this})(s) * ((\mathsf{r} = \mathbf{null} * \triangleright\ulcorner\mathsf{Q}(\mathbf{this}, \mathbf{null})\urcorner) \vee (\mathsf{r} \neq \mathbf{null} * \ulcorner lst_r(\mathsf{r}, _)\urcorner * \overline{\mathsf{P}(\mathbf{this})}))\rangle
\end{array}
$$

$\langle \mathsf{r}.\ \mathsf{region}(\{s\}, T, I(\mathbf{this}), \mathsf{r}) * ((\mathsf{r} = \mathbf{null} * \triangleright\ulcorner \mathsf{Q}(\mathbf{this}, \mathbf{null})\urcorner) \vee (\mathsf{r} \neq \mathbf{null} * \ulcorner lst_r(\mathsf{r}, _)\urcorner * \overline{\mathsf{P}(\mathbf{this})}))\rangle$
$\{\mathsf{r}.\ stack(\mathbf{this}) * ((\mathsf{r} = \mathbf{null} * \triangleright\ulcorner \mathsf{Q}(\mathbf{this}, \mathbf{null})\urcorner) \vee (\mathsf{r} \neq \mathbf{null} * \ulcorner lst_r(\mathsf{r}, _)\urcorner * \overline{\mathsf{P}(\mathbf{this})}))\}$

$[stack(\mathbf{this}) * \ulcorner \mathsf{R}\urcorner * \overline{\mathsf{P}(\mathbf{this})}]$
$\langle stack(\mathbf{this}) * \ulcorner \mathsf{R}\urcorner * \overline{\mathsf{P}(\mathbf{this})}\rangle$
$\langle \mathsf{region}(\{s\}, T, I(\mathbf{this}), \mathsf{r}) * \ulcorner \mathsf{R}\urcorner * \overline{\mathsf{P}(\mathbf{this})}\rangle$
$\langle \triangleright I(\mathbf{this})(s) * \ulcorner \mathsf{R}\urcorner * \overline{\mathsf{P}(\mathbf{this})}\rangle$
$\langle \triangleright(\ulcorner \mathbf{this}.\mathsf{head} \mapsto \mathsf{z} * lst_r(\mathsf{z}, \alpha) * \triangleright \mathbf{this}_{\mathsf{cont}} \xrightarrow{1/2} \alpha\urcorner) * \ulcorner \mathsf{R}\urcorner * \overline{\mathsf{P}(\mathbf{this})}\rangle$
  CAS(**this**.head, nHead, oHead)
$\langle \mathsf{r}.\ (\mathsf{r} = \mathbf{true} * \mathsf{z} = \mathsf{oHead} * \ulcorner \mathbf{this}.\mathsf{head} \mapsto \mathsf{nHead} * lst_r(\mathsf{z}, \alpha) * \triangleright \mathbf{this}_{\mathsf{cont}} \xrightarrow{1/2} \alpha * \mathsf{R} * \mathsf{P}(\mathbf{this})\urcorner) \vee$
$\quad (\mathsf{r} = \mathbf{false} * \mathsf{z} \neq \mathsf{oHead} * \ulcorner \mathbf{this}.\mathsf{head} \mapsto \mathsf{z} * lst_r(\mathsf{z}, \alpha) * \triangleright \mathbf{this}_{\mathsf{cont}} \xrightarrow{1/2} \alpha * \mathsf{R}\urcorner * \overline{\mathsf{P}(\mathbf{this})})\rangle$
$\langle \mathsf{r}.\ (\mathsf{r} = \mathbf{true} * \ulcorner \mathbf{this}.\mathsf{head} \mapsto \mathsf{nHead} * \triangleright \mathbf{this}_{\mathsf{cont}} \xrightarrow{1/2} (\mathsf{y} :: \beta) * lst_r(\mathsf{nHead}, \beta) * \mathsf{P}(\mathbf{this}) * \mathsf{oHead.val} \Mapsto \mathsf{y}\urcorner) \vee$
$\quad (\mathsf{r} = \mathbf{false} * I(\mathbf{this})(s) * \overline{\mathsf{P}(\mathbf{this})})\rangle$
$\langle \mathsf{r}.\ (\mathsf{r} = \mathbf{true} * \ulcorner \mathbf{this}.\mathsf{head} \mapsto \mathsf{nHead} * \triangleright \mathbf{this}_{\mathsf{cont}} \xrightarrow{1/2} \beta * lst_r(\mathsf{nHead}, \beta) * \triangleright \mathsf{Q}(\mathbf{this}, \mathsf{y}) * \mathsf{oHead.val} \Mapsto \mathsf{y}\urcorner) \vee$
$\quad (\mathsf{r} = \mathbf{false} * I(\mathbf{this})(s) * \overline{\mathsf{P}(\mathbf{this})})\rangle$
$\langle \mathsf{r}.\ (\mathsf{r} = \mathbf{true} * I(\mathbf{this})(s) * \ulcorner \triangleright \mathsf{Q}(\mathbf{this}, \mathsf{y}) * \mathsf{oHead.val} \Mapsto \mathsf{y}\urcorner) \vee (\mathsf{r} = \mathbf{false} * I(\mathbf{this})(s) * \overline{\mathsf{P}(\mathbf{this})})\rangle$
$\langle \mathsf{r}.\ \mathsf{region}(\{s\}, T, I(\mathbf{this}), \mathsf{r}) * (\mathsf{r} = \mathbf{true} * \ulcorner \triangleright \mathsf{Q}(\mathbf{this}, \mathsf{y}) * \mathsf{oHead.val} \Mapsto \mathsf{y}\urcorner) \vee (\mathsf{r} = \mathbf{false} * \overline{\mathsf{P}(\mathbf{this})})\rangle$
$[\mathsf{r}.\ stack(\mathbf{this}) * ((\mathsf{r} = \mathbf{true} * \ulcorner \triangleright \mathsf{Q}(\mathbf{this}, \mathsf{y}) * \mathsf{oHead.val} \Mapsto \mathsf{y}\urcorner) \vee (\mathsf{r} = \mathbf{false} * \overline{\mathsf{P}(\mathbf{this})}))]$

where R is shorthand for $\mathsf{oHead.val} \Mapsto \mathsf{y} * \mathsf{oHead.next} \Mapsto \mathsf{nHead} * lst_r(\mathsf{nHead}, _)$.

7.4 Deriving an SC shared bag specification

In this section illustrate how to *derive* an SC specification for a shared bag from the abstract specification of Treiber's stack. The SC shared bag specification is given below and allows clients to associate ownership of a resource with each element in the bag through an SC assertion P:

$$\begin{aligned}
&\exists bag : (\mathsf{Val} \to \mathsf{Prop_{SC}}) \times \mathsf{Val} \to \mathsf{Prop_{SC}}.\\
&\quad \forall \mathsf{P} : \mathsf{Val} \to \mathsf{Prop_{SC}}.\ \forall x : \mathsf{Val}.\ \mathsf{stable}(\mathsf{P}(x)) \Rightarrow\\
&\qquad\quad \{emp\}\ \mathsf{Stack}()\ \{\mathsf{r}.\ bag(\mathsf{P}, \mathsf{r})\}\\
&\qquad \wedge\ \{bag(\mathsf{P}, \mathbf{this}) * \mathsf{P}(\mathsf{x})\}\ \mathsf{Stack.push}(\mathsf{x})\ \{emp\}\\
&\qquad \wedge\ \{bag(\mathsf{P}, \mathbf{this})\}\ \mathsf{Stack.pop}()\ \{\mathsf{r}.\ \mathsf{r} = \mathsf{null} \vee \mathsf{P}(\mathsf{r})\}\\
&\qquad \wedge\ \mathsf{valid}(\forall \mathsf{x} : \mathsf{Val}.\ bag(\mathsf{P}, \mathsf{x}) \Leftrightarrow bag(\mathsf{P}, \mathsf{x}) * bag(\mathsf{P}, \mathsf{x}))\\
&\qquad \wedge\ \forall \mathsf{x} : \mathsf{Val}.\ \mathsf{stable}(bag(\mathsf{P}, \mathsf{x}))
\end{aligned}$$

To derive this specification we introduce another shared region that owns the other half of the phantom-field containing the abstract state of the underlying stack *and* the resources associated with each element currently in the underlying stack. We thus define *bag* as

follows:

$$bag(\mathsf{P},\mathsf{x}) = \exists \mathsf{r}_1, \mathsf{r}_2 : \mathsf{RId}.\ \mathsf{region}(\{s\}, T_{\mathrm{bag}}, I_{\mathrm{bag}}(\mathsf{P},\mathsf{x}), \mathsf{r}_1) * stack(\mathsf{r}_2, \mathsf{x}) * \mathsf{r}_1 < \mathsf{r}_2$$

where T_{bag} is a labelled transition system with a single state s and I_{bag} is defined as follows:

$$I_{\mathrm{bag}}(\mathsf{P},\mathsf{x})(s) = \exists \alpha : \mathrm{seq}\ \mathsf{Val}.\ \ulcorner \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \alpha * (\circledast_{\mathsf{y} \in mem(\alpha)} \mathsf{P}(\mathsf{y}))\urcorner$$

For the $\mathsf{Stack.push}$ method we thus have to show that

$$\forall \alpha : \mathrm{seq}\ \mathsf{Val}.\ \forall \mathsf{x}, \mathsf{y}.$$
$$\mathsf{region}(\{s\}, T_{\mathrm{bag}}, I_{\mathrm{bag}}(\mathsf{P},\mathsf{x}), \mathsf{r}_1) * \ulcorner \mathsf{P}(\mathsf{y})\urcorner * \triangleright \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \alpha \sqsubseteq^{\mathsf{RId} \setminus \{\mathsf{r}_2\}} \triangleright \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} (\mathsf{y} :: \alpha)$$

by applying the region opening rule, this reduces to proving that

$$\exists \beta : \mathrm{seq}\ \mathsf{Val}.\ \triangleright \ulcorner \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \beta * (\circledast_{\mathsf{y} \in mem(\beta)} \mathsf{P}(\mathsf{y}))\urcorner * \ulcorner \mathsf{P}(\mathsf{y})\urcorner * \triangleright \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \alpha \sqsubseteq^{\mathsf{RId} \setminus \{\mathsf{r}_1, \mathsf{r}_2\}}$$
$$\exists \beta : \mathrm{seq}\ \mathsf{Val}.\ \ulcorner \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \beta * (\circledast_{\mathsf{y} \in mem(\beta)} \mathsf{P}(\mathsf{y}))\urcorner * \triangleright \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} (\mathsf{y} :: \alpha)$$

This is easily seen to hold, as $\mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \alpha * \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \beta \Rightarrow \alpha = \beta$ and

$$\triangleright \mathsf{x}_{\mathsf{cont}} \mapsto \alpha \sqsubseteq \triangleright \mathsf{x}_{\mathsf{cont}} \mapsto \beta$$

Likewise, for the $\mathsf{Stack.pop}$ method we have to show that

$$\forall \mathsf{x} : \mathsf{Val}.$$
$$\mathsf{region}(\{s\}, T_{\mathrm{bag}}, I_{\mathrm{bag}}(\mathsf{P},\mathsf{x}), \mathsf{r}_1) * \triangleright \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \varepsilon \sqsubseteq^{\mathsf{RId} \setminus \{\mathsf{r}_2\}} \triangleright (\mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \varepsilon * \ulcorner (\mathbf{null} = \mathbf{null} \vee \mathsf{P}(\mathbf{null}))\urcorner)$$

and

$$\forall \alpha : \mathrm{seq}\ \mathsf{Val}.\ \forall \mathsf{x}, \mathsf{y} : \mathsf{Val}.$$
$$\mathsf{region}(\{s\}, T_{\mathrm{bag}}, I_{\mathrm{bag}}(\mathsf{P},\mathsf{x}), \mathsf{r}_1) * \triangleright \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} (\mathsf{y} :: \alpha) \sqsubseteq^{\mathsf{RId} \setminus \{\mathsf{r}_2\}} \triangleright (\mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \alpha * \ulcorner (\mathsf{y} = \mathbf{null} \vee \mathsf{P}(\mathsf{y}))\urcorner)$$

This first proof obligation holds trivially, as $\mathbf{null} = \mathbf{null}$. To discharge the second proof obligation, we apply the region opening rule, which leaves us with the following proof obligation:

$$\exists \beta : \mathrm{seq}\ \mathsf{Val}.\ \ulcorner \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \beta * (\circledast_{\mathsf{y} \in mem(\beta)} \mathsf{P}(\mathsf{y})\urcorner) * \triangleright \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} (\mathsf{y} :: \alpha)\ \sqsubseteq^{\mathsf{RId} \setminus \{\mathsf{r}_1, \mathsf{r}_2\}}$$
$$\exists \beta : \mathrm{seq}\ \mathsf{Val}.\ \ulcorner \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \beta * (\circledast_{\mathsf{y} \in mem(\beta)} \mathsf{P}(\mathsf{y})\urcorner) * \triangleright \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \alpha * \triangleright \ulcorner \mathsf{P}(\mathsf{y})\urcorner$$

Again, this follows easily since $\mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} (\mathsf{y} :: \alpha) * \mathsf{x}_{\mathsf{cont}} \overset{1/2}{\longmapsto} \beta \Rightarrow (\mathsf{y} :: \alpha) = \beta$.

8 Verification of a bounded ticket lock in the TSO logic

In this section we verify a bounded ticket lock [4, 1] implementation using the TSO logic (see [9] for a proof in the case of C11 acquire/release).

8.1 Specification

A ticket lock is a fair locking algorithm where threads obtain a ticket number and wait for it to be served. In a bounded ticket lock with bound b, the ticket number goes back to 0 when it reaches b, which means that it can be used by at most b clients. In TSO, the increment to the serving number in the release can be buffered as for the spinlock (see Section 6).
We thus give it a lock specification with a bounded number of clients, where we separate obtaining a ticket from checking whether it is being served. To get the strong specification where the lock invariant holds in main memory, a thread can have at most one lock permission. Our proof carries over to the weaker invariant where the lock invariant holds only from the perspective of the locking thread. Note that the locking permissions have to be attached to threads, because their transmission on a side-channel that would bypass the TSO FIFO policy would cause violations of the specification.

$$
\begin{array}{l}
\forall \mathsf{b} : \mathbb{N}.\ 0 < \mathsf{b} \Longrightarrow \\
\exists \mathit{mayTicket}, \mathit{mayRel} : \mathsf{Val} \times \mathsf{Prop_{SC}} \times \mathrm{TId} \to \mathsf{Prop_{SC}}. \\
\exists \mathit{mayLock} : \mathsf{Val} \times \mathsf{Prop_{SC}} \times \mathrm{TId} \times \mathsf{Val} \to \mathsf{Prop_{SC}}. \\
\quad \forall \mathsf{R} : \mathsf{Prop_{SC}}.\ \mathsf{stable}(\mathsf{R}) \ \Rightarrow \\
\qquad\quad \forall T : \mathrm{set\,TId}.\ [\overline{\mathsf{R}} * 0 < \mathsf{b} * |T| = \mathsf{b}]\ \mathsf{TicketLock(b)}\ [\mathsf{r}.\ \circledast_{t \in T}\ \mathit{mayTicket}(\mathsf{r}, \mathsf{R}, t)] \\
\qquad \wedge\ [\mathit{mayTicket}(\textbf{this}, \mathsf{R}, t) * \mathsf{iam}(t)]\ \mathsf{TicketLock.ticket()}\ [r.\ \mathit{mayLock}(\textbf{this}, \mathsf{R}, t, r) * \ulcorner \mathsf{R} \urcorner] \\
\qquad \wedge\ [\mathit{mayLock}(\textbf{this}, \mathsf{R}, t, \mathsf{y}) * \mathsf{iam}(t)]\ \mathsf{TicketLock.lock(y)}\ [r.\ \mathrm{if}\ r\ \mathrm{then}\ (\mathit{mayRel}(\textbf{this}, \mathsf{R}) * \ulcorner \mathsf{R} \urcorner)\ \mathrm{else}\ (\mathit{mayLoc} \\
\qquad \wedge\ [\mathit{mayRel}(\textbf{this}, \mathsf{R}, t) * \overline{\mathsf{R}} * \mathsf{iam}(t)]\ \mathsf{TicketLock.release()}\ [\mathit{mayTicket}(\textbf{this}, \mathsf{R}, t)] \\
\qquad \wedge\ \forall \mathsf{x} : \mathsf{Val}.\ \forall t : \mathrm{TId}.\ \mathsf{stable}(\mathit{mayTicket}(\mathsf{x}, \mathsf{R}, t)) \\
\qquad \wedge\ \forall \mathsf{x}, \mathsf{y} : \mathsf{Val}.\ \forall t : \mathrm{TId}.\ \mathsf{stable}(\mathit{mayLock}(\mathsf{x}, \mathsf{R}, t, \mathsf{y})) \\
\qquad \wedge\ \forall \mathsf{x} : \mathsf{Val}.\ \forall t : \mathrm{TId}.\ \mathsf{stable}(\mathit{mayRel}(\mathsf{x}, \mathsf{R}, t))
\end{array}
$$

```
class TicketLock {
  int bound;
  int ticket;
  int serving;

  TicketLock(int bound) =
   (this.bound := bound;
```

```
    this.ticket := 0;
    this.serving := 0;
    fence;
    this)

  int ticket() =
    let b = this.bound in
    let y = FAI(ref this.ticket, b) in
    y

  bool lock(int y) =
    let s = this.serving in
    s = y

  unit release() =
    let b = this.bound in
    let s = this.serving in
    this.serving := (s + 1) % b
}
```

8.2 Discussion of the design choices

In the specification above, a *mayTicket* permission is usable only by a particular thread, fixed at the creation of the ticket lock. Moreover, each thread can have at most one *mayTicket* permission. ticket and spin.

The restriction that a *mayTicket* permission is usable only by a particular thread is crucial to the proof. The logic does (on purpose) not exclude the existence of "side channels" on which permissions could be passed. Such a side channels could be used as follows (with the bound $b = 2$):

1. initially, $\ulcorner x.\mathsf{serving} \mapsto 0 \urcorner * \ulcorner x.\mathsf{ticket} \mapsto 0 \urcorner$ ($\mathsf{U}_{0,0}$)
2. t1 takes a ticket, so t1's y is 0, and $\ulcorner x.\mathsf{serving} \mapsto 0 \urcorner * \ulcorner x.\mathsf{ticket} \mapsto 1 \urcorner$ ($\mathsf{U}_{1,1}$)
3. t2 takes a ticket, so t2's y is 1, and $\ulcorner x.\mathsf{serving} \mapsto 0 \urcorner * \ulcorner x.\mathsf{ticket} \mapsto 0 \urcorner$ ($\mathsf{U}_{0,2}$)
4. t1 locks (indeed, t1's y is 0, and $\ulcorner x.\mathsf{serving} \mapsto 0 \urcorner$), and $\ulcorner x.\mathsf{serving} \mapsto 0 \urcorner * \ulcorner x.\mathsf{ticket} \mapsto 0 \urcorner$ ($\mathsf{L}_{0,2}$)
5. t1 unlocks (but does not flush yet), so $(\ulcorner x.\mathsf{serving} \mapsto 0 \urcorner \ \mathcal{U}_{t1} \ \ulcorner x.\mathsf{serving} \mapsto 1 \urcorner) * \ulcorner x.\mathsf{ticket} \mapsto 0 \urcorner$ ($\mathsf{U}_{1,1}$)
6. t1 sends its *mayTicket* permission to t3 over a side-channel, so that it reaches t3 before the $\ulcorner x.\mathsf{serving} \mapsto 1 \urcorner$ in t1's buffer reaches main memory

7. t3, using t1's $mayTicket$, takes a ticket, so t3's y is 0, and $(\ulcorner x.\mathsf{serving} \mapsto 0\urcorner\ \mathcal{U}_{t1}$ $\ulcorner x.\mathsf{serving} \mapsto 1\urcorner) * \ulcorner x.\mathsf{ticket} \mapsto 1\urcorner$ $(\mathsf{U}_{1,2})$
8. t3 locks, when it was t2's turn, and does not have $\ulcorner R\urcorner$, as R can still be in t1's buffer.

The restriction that each thread has at most one $mayTicket$ permission is required to ensure that the lock invariant holds in main memory. It can be lifted (and the proof carries over, except that there can be a chain of updates rather than a single one) if the specification only requires the lock invariant to hold from the point of view of the thread holding the lock. Without it, lock would not be able to ensure $\ulcorner R\urcorner$, only $\ulcorner R\ \mathbf{in}\ t\urcorner$. A thread with multiple $mayTicket$ permissions could execute ticket ticketlock release lock, after which there is no guarantee that R holds in main memory (because there is not interposing FAI).

8.3 acquire version

It is possible to recover a version that tickets and locks in one operation (as used in [9]):

...

$\forall \mathsf{R} : \mathsf{Prop}_{\mathsf{sc}}.\ \mathsf{stable}(\mathsf{R}) \Rightarrow$

...

$\wedge\ [mayTicket(\mathbf{this}, \mathsf{R}, t) * \mathsf{iam}(t)]\ \mathsf{TicketLock.acquire}()\ [r.\ mayRel(\mathbf{this}, \mathsf{R}, t, r) * \ulcorner \mathsf{R}\urcorner * \mathsf{iam}(t)]$

...

```
unit acquire() =
  let y = ticket() in
  acquire_aux(y)

unit acquire_aux(int y) =
  if lock(y) then ()
  else acquire_aux(y)

unit acquire() =
  [mayTicket(this, R, t) ∗ iam(t)]
  let y = ticket() in
  [mayLock(this, R, t, y) ∗ iam(t)]
  acquire_aux(y)
  [mayRel(this, R, t) ∗ ⌜R⌝]

unit acquire_aux(int y) =
  [mayLock(this, R, t, y) ∗ iam(t)]
  if lock(y) then [mayRel(this, R, t) ∗ ⌜R⌝] ()
```

else $[\mathit{mayLock}(\textbf{this}, R, t, \mathsf{y}) * \mathsf{iam}(t)]$ acquire_aux(y)
$[\mathit{mayRel}(\textbf{this}, R, t) * \ulcorner R \urcorner]$

8.4 Definitions

The region has to maintain the information of which threads have permissions to ticket, and which threads have pending tickets, in which order. The region maintains this information by elimination:

- For each thread that never had a ticket permission, or gave it away temporarily to get a ticket (and is in the list of threads with pending tickets), the region holds the ticket permission for that thread.
- For each ticket number, for each thread that doesn't have that ticket number, the region holds the ticket permission for that ticket for that thread; combined with the previous point, this allows the region to deduce which threads have pending tickets.
- For each thread that doesn't hold the lock, it holds the permission for that thread to hold the lock (this complements the previous point: the thread doesn't have its ticket permission yet, its ticket is still somehow pending).

$$\begin{aligned}
\mathit{mayTicket}(x, R, t) &= \exists r : \mathsf{RId}, b : \mathbb{N}.\ 0 < b * \ulcorner x.\mathsf{bound} \Mapsto b \urcorner * \mathsf{region}(U \cup L, \mathsf{T}_{\mathrm{lock}}, I_{\mathrm{lock}}(x, R, b, r), r) * [\mathrm{T}_t]^r_1 \\
\mathit{mayLock}(x, R, t, y) &= \exists r : \mathsf{RId}.b : \mathbb{N}.\ 0 < b * \ulcorner x.\mathsf{bound} \Mapsto b \urcorner * \mathsf{region}(S_y, \mathsf{T}_{\mathrm{lock}}, I_{\mathrm{lock}}(x, R, b, r), r) * [\mathrm{L}^t_y]^r_1 \\
\mathit{mayRel}(x, R, t) &= \exists r : \mathsf{RId}, b : \mathbb{N}.\ 0 < b * \ulcorner x.\mathsf{bound} \Mapsto b \urcorner * \mathsf{region}(L, \mathsf{T}_{\mathrm{lock}}, I_{\mathrm{lock}}(x, R, b, r), r) * [\mathrm{U}_t]^r_1
\end{aligned}$$

States of the protocol:

$$\{\mathsf{U}_{n,m} | 0 \le n < b \wedge 0 \le m \le b\} \cup \{\mathsf{L}_{n,m} | 0 \le n < b \wedge 1 \le m \le b\}$$

Transitions of the protocol:

$$\begin{aligned}
\mathrm{T}_t &: \mathsf{U}_{n,m} \to \mathsf{U}_{n,m+1} && \text{if } m < b \\
&: \mathsf{L}_{n,m} \to \mathsf{L}_{n,m+1} && \text{if } m < b \\
\mathrm{L}^t_n &: \mathsf{U}_{n,m} \to \mathsf{L}_{n,m} && \text{if } 0 < m \\
\mathrm{U}_t &: \mathsf{L}_{n,m} \to \mathsf{U}_{(n+1)\%b,m-1} && \text{if } 0 < m
\end{aligned}$$

Interesting sets of states:

$$\begin{aligned}
U &= \{\mathsf{U}_{n,m} | 0 \le n < b \wedge 0 \le m \le b\} \\
L &= \{\mathsf{L}_n | 0 \le n < b \wedge 1 \le m \le b\} \\
L_s &= \{\mathsf{L}_{s,m} | 1 \le m \le b\} \\
S_y &= \begin{array}{l} \{\mathsf{L}_{n,m} | 0 \le n < b \wedge (\exists i.\ 0 < i < m \wedge (n+i)\%b = y)\} \\ \cup \{\mathsf{U}_{n,m} | 0 \le n < b \wedge (\exists i.\ 0 \le i < m \wedge (n+i)\%b = y)\} \end{array}
\end{aligned}$$

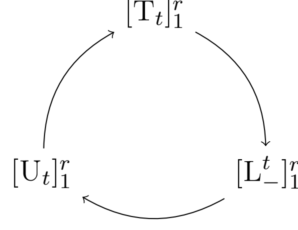


Figure 7: Permission life cycle

Interpretation:

$$
\begin{aligned}
I_{\text{lock}}(x,R,b,r)(\mathsf{U}_{n,m}) = {} & \exists \hat{T} : \text{set TId}.\ \exists T : \text{seq TId}.\ |\hat{T}| = b * |T|_\ell = m * \text{norepeat}(T) * \text{ran}(T) \subseteq \hat{T} \\
& * ((\exists t' : \text{TId}.\ t' \notin_\ell T.\ (\!|(\ulcorner x.\mathsf{serving} \mapsto (n-1)\%b \urcorner * \ulcorner R \textbf{ in } t' \urcorner)\ \mathcal{U}_{t'}\ \ulcorner x.\mathsf{serving} \mapsto n * R \urcorner|\!)) \\
& \quad \vee \ulcorner x.\mathsf{serving} \mapsto n * R \urcorner) \\
& * \ulcorner x.\mathsf{ticket} \mapsto (n+m)\%b \urcorner \\
& * (\circledast_{t \in (\text{TId} \setminus (\hat{T} \setminus \text{ran}(T)))} [\mathrm{T}_t]^r_1) * (\circledast_{t \in \text{TId}} [\mathrm{U}_t]^r_1) \\
& * \circledast_{n+m \le i < n+b} (\circledast_{t \in \text{TId}} [\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+b} (\circledast_{t \in \text{TId} \setminus \{T[i-(n+b)]\}} [\mathrm{L}^t_{i\%b}]^r_1) \\
I_{\text{lock}}(x,R,b,r)(\mathsf{L}_{n,m}) = {} & \exists \hat{T} : \text{set TId}.\ \exists T : \text{seq TId}.\ |\hat{T}| = b * |T|_\ell = m * \text{norepeat}(T) * \text{ran}(T) \subseteq \hat{T} \\
& * \ulcorner x.\mathsf{serving} \mapsto n \urcorner * \ulcorner x.\mathsf{ticket} \mapsto (n+m)\%b \urcorner \\
& * (\circledast_{t \in (\text{TId} \setminus (\hat{T} \setminus \text{ran}(T)))} [\mathrm{T}_t]^r_1) * (\circledast_{t \in \text{TId} \setminus \{T[0]\}} [\mathrm{U}_t]^r_1) \\
& * \circledast_{n+m \le i < n+b+1} (\circledast_{t \in \text{TId}} [\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b+1 \le i < n+m+b} (\circledast_{t \in \text{TId} \setminus \{\text{tl}(T)[i-(n+b+1)]\}} [\mathrm{L}^t_{i\%b}]^r_1)
\end{aligned}
$$

For example, in $\mathsf{L}_{1,3}$, the region is missing a $[\mathrm{L}^t_2]^r_1$ and a $[\mathrm{L}^{t'}_3]^r_1$, but has the $[\mathrm{L}^t_1]^r_1$ (which $\mathsf{U}_{1,3}$ doesn't).

Permissions have a form of life cycle (see Fig. 7), and the region holds all the permissions (for all $t \in \text{TId}$) except

- in the unlocked state, $(\circledast_{t \in \hat{T} \setminus \text{ran}(T)} [\mathrm{T}_t]^r_1) * (\circledast_{t \in_\ell T} [\mathrm{L}^t_{-}]^r_1)$
- in the locked state, $(\circledast_{t \in \hat{T} \setminus \text{ran}(T)} [\mathrm{T}_t]^r_1) * (\circledast_{t \in_\ell \text{tl}(T)} [\mathrm{L}^t_{-}]^r_1) * [\mathrm{U}_{T[0]}]^r_1$

As T is quantified over existentially, to make sure it is "the right T", it has to be related to the threads it should talk about by "permission accounting"; for example, as the region holds $(\circledast_{t \in \text{TId} \setminus \{T[0]\}} [\mathrm{U}_t]^r_1)$, the thread that holds the lock has to be $T[0]$. The same trick is used for $\hat{T}$.

8.5 Proof outline

8.5.1 Outline

```
TicketLock(int bound) =
 (this.bound := bound;
```

```
    this.ticket := 0;
    this.serving := 0;
    fence;
    this)

  int ticket() =
    [region(U ∪ L, T_lock, I_lock(this, R, b, r), r) ∗ [T_t]^r_1 ∗ iam(t)]
    let b = this.bound in
    [region(U ∪ L, T_lock, I_lock(this, R, b, r), r) ∗ [T_t]^r_1 ∗ iam(t)]
    let y = FAI(ref this.ticket, b) in
    [region(S_y, T_lock, I_lock(this, R, b, r), r) ∗ [L^t_y]^r_1 ∗ iam(t)]
    y

  bool lock(int y) =
    [region(S_y, T_lock, I_lock(this, R, b, r), r) ∗ [L^t_y]^r_1 ∗ iam(t)]
    let s = this.serving in
    [(if s = y then region(L, T_lock, I_lock(this, R, b, r), r) ∗ (⊛_{t∈TId}[U_t]^r_1) ∗ ⌜R⌝
    else (region(S_y, T_lock, I_lock(this, R, b, r), r) ∗ [L^t_y]^r_1)) ∗ iam(t)]
    s = y
    [v. (if v then (region(L, T_lock, I_lock(this, R, b, r), r) ∗ (⊛_{t∈TId}[U_t]^r_1) ∗ ⌜R⌝)
    else (region(S_y, T_lock, I_lock(this, R, b, r), r) ∗ [L^t_y]^r_1)) ∗ iam(t)]

  unit release() =
    [region(L, T_lock, I_lock(this, R, b, r), r) ∗ [U_t]^r_1 ∗ R̄ ∗ iam(t)]
    let b = this.bound in
    [region(L, T_lock, I_lock(this, R, b, r), r) ∗ [U_t]^r_1 ∗ R̄ ∗ iam(t)]
    let s = this.serving in
    [region(L_s, T_lock, I_lock(this, R, b, r), r) ∗ [U_t]^r_1 ∗ R̄ ∗ iam(t)]
    this.serving := (s + 1) % b
    [region(U ∪ L, T_lock, I_lock(this, R, b, r), r) ∗ [T_t]^r_1 ∗ iam(t)]
```

8.5.2 Discussion

1. The treatment of the bound b is no surprise, and is not shown in the rest of the proof to minimise clutter.

   [$\exists b : \mathbb{N}.\ 0 < b * \ulcorner$**this**.bound $\mapsto b\urcorner * \ldots$]
   let b = **this**.bound **in**
   [$0 <$ b $* \ulcorner$**this**.bound $\mapsto$ b$\urcorner * \ldots$]

2. The iam(t) are threaded through and duplicated as needed for the reads and writes, but this is not shown in the rest of the proof to minimise clutter.

3. The FAI in ticket exchanges the ticket permission for a lock permission, and constrains the set of states the protocol to be those states where the obtained ticket

number is registered as waiting to be served. From the constraint on the set of states and the lock permission, it is possible to deduce that the protocol is in a state where the *thread* is registered as waiting to be served this ticket number.

$[\mathsf{region}(U \cup L, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, \mathsf{b}, r), r) * [\mathrm{T}_t]^r_1 * \mathsf{iam}(t)]$

let y = FAI(ref **this**.ticket, b) **in**

$[\mathsf{region}(S_{\mathsf{y}}, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, \mathsf{b}, r), r) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t)]$

The treatment of b is no surprise, and is not shown in the rest of the proof to minimise clutter.

4. The read of serving in lock can end in two ways: if the value read is different from the ticket number y, nothing changes; however, if serving is equal to the ticket number, then it is that thread's turn to be served, so the lock permission is exchanged for an unlock permission and the resource, and the state is now a locked state.

   $[\mathsf{region}(S_{\mathsf{y}}, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, b, r), r) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t)]$

   let s = **this**.serving **in**

   $[(\mathsf{if\ s = y\ then\ } (\mathsf{region}(L, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, b, r), r) * [\mathrm{U}_t]^r_1 * \ulcorner R \urcorner)$
   $\mathsf{else\ } (\mathsf{region}(S_{\mathsf{y}}, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, b, r), r) * [\mathrm{L}^t_{\mathsf{y}}]^r_1)) * \mathsf{iam}(t)]$

5. The read of serving in release restricts the set of states. This could have been done at the end of lock, but then the ticket number would have needed to be threaded to the release, which feels unnatural. The write to serving exchanges the unlock permission and the (locally-holding) resource for a ticket permission, and loses any control over the state of the protocol (although it is still possible to constrain the state of the protocol from the ticket permission, it is not needed). Because of the FIFO behaviour of TSO buffers, by the time the increment to serving has propagated to main memory, the resource holds in main memory as well.

   $[\mathsf{region}(L, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, \mathsf{b}, r), r) * [\mathrm{U}_t]^r_1 * \overline{R} * \mathsf{iam}(t)]$

   let s = **this**.serving **in**

   $[\mathsf{region}(L_{\mathsf{s}}, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, \mathsf{b}, r), r) * [\mathrm{U}_t]^r_1 * \overline{R} * \mathsf{iam}(t)]$

   this.serving := (s + 1) % b

   $[\mathsf{region}(U \cup L, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, \mathsf{b}, r), r) * [\mathrm{T}_t]^r_1 * \mathsf{iam}(t)]$

   Note that immediately after the increment to serving, the state of the region is a bit ambiguous: it is morally unlocked from the point of view of the releasing thread, but morally still locked from the point of view of the other threads (until the increment reaches main memory); however, from the point of view of the logic, the region immediately becomes unlocked, because at this point of the proof, the possibility of the *mayTicket* permission being passed to another thread on a side channel has to be considered (which would be a problem if the *mayTicket* permissions were not attached to threads).

8.6 Proof

8.6.1 ticket

1. case U:

$[\mathrm{region}(U \cup L, \mathsf{T_{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, b, r), r) * [\mathrm{T}_t]^r_1 * \mathsf{iam}(t)]$

$\langle \triangleright I_{\mathsf{lock}}(\mathbf{this}, R, b, n)(\mathsf{U}_{n,m}) * [\mathrm{T}_t]^r_1 * \mathsf{iam}(t)\rangle$

$\langle \triangleright(\exists \hat{T} : \mathrm{set\ TId}.\ \exists T : \mathrm{seq\ TId}.\ |\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * ((\exists t' : \mathrm{TId}.\ t' \notin_\ell T * ((\ulcorner \mathbf{this}.\mathsf{serving} \mapsto (n-1)\%b \urcorner * \ulcorner R \textbf{ in } t' \urcorner)\ \mathcal{U}_{t'}\ \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner)) \vee \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner) * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in \mathrm{TId}} [\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))} [\mathrm{T}_t]^r_1) * \circledast_{n+m \le i < n+b} (\circledast_{t \in \mathrm{TId}} [\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+b} (\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}} [\mathrm{L}^t_{i\%b}]^r_1)) * [\mathrm{T}_t]^r_1 * \mathsf{iam}(t)\rangle$

$\langle \triangleright(|\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * ((\exists t' : \mathrm{TId}.\ t' \notin_\ell T * ((\ulcorner \mathbf{this}.\mathsf{serving} \mapsto (n-1)\%b \urcorner * \ulcorner R \textbf{ in } t' \urcorner)\ \mathcal{U}_{t'}\ \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner)) \vee \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner) * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in \mathrm{TId}} [\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))} [\mathrm{T}_t]^r_1) * \circledast_{n+m \le i < n+b} (\circledast_{t \in \mathrm{TId}} [\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+b} (\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}} [\mathrm{L}^t_{i\%b}]^r_1)) * [\mathrm{T}_t]^r_1 * \mathsf{iam}(t)\rangle$

$\langle \triangleright(|\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * ((\exists t' : \mathrm{TId}.\ t' \notin_\ell T * ((\ulcorner \mathbf{this}.\mathsf{serving} \mapsto (n-1)\%b \urcorner * \ulcorner R \textbf{ in } t' \urcorner)\ \mathcal{U}_{t'}\ \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner)) \vee \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner) * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in \mathrm{TId}} [\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T@[t])))} [\mathrm{T}_t]^r_1) * \circledast_{n+m \le i < n+b} (\circledast_{t \in \mathrm{TId}} [\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+b} (\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}} [\mathrm{L}^t_{i\%b}]^r_1)) * \mathsf{iam}(t)\rangle$

$\langle \triangleright(|\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * (((\exists t' : \mathrm{TId}.\ t' \notin_\ell T * (\ulcorner \mathbf{this}.\mathsf{serving} \mapsto (n-1)\%b \urcorner * \ulcorner R \textbf{ in } t' \urcorner)\ \mathcal{U}_{t'}\ \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner) \vee (\exists t' : \mathrm{TId}. t' \notin_\ell T * \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner)) \vee \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner) * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in \mathrm{TId}} [\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T@[t])))} [\mathrm{T}_t]^r_1) * \circledast_{n+m \le i < n+b} (\circledast_{t \in \mathrm{TId}} [\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+b} (\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}} [\mathrm{L}^t_{i\%b}]^r_1)) * \mathsf{iam}(t)\rangle$

$\langle \triangleright(|\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * (((\exists t' : \mathrm{TId}.\ t' \notin_\ell T * (\ulcorner \mathbf{this}.\mathsf{serving} \mapsto (n-1)\%b \urcorner * \ulcorner R \textbf{ in } t' \urcorner)\ \mathcal{U}_{t'}\ \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner) \vee (\ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner)) \vee \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner) * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in \mathrm{TId}} [\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T@[t])))} [\mathrm{T}_t]^r_1) * \circledast_{n+m \le i < n+b} (\circledast_{t \in \mathrm{TId}} [\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+b} (\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}} [\mathrm{L}^t_{i\%b}]^r_1)) * \mathsf{iam}(t)\rangle$

$\langle \triangleright(|\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * (\exists t' : \mathrm{TId}.\ t' \notin_\ell T * (\ulcorner \mathbf{this}.\mathsf{serving} \mapsto (n-1)\%b \urcorner * \ulcorner R \textbf{ in } t' \urcorner)\ \mathcal{U}_{t'}\ \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner \vee \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner) * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in \mathrm{TId}} [\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T@[t])))} [\mathrm{T}_t]^r_1) * \circledast_{n+m \le i < n+b} (\circledast_{t \in \mathrm{TId}} [\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+b} (\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}} [\mathrm{L}^t_{i\%b}]^r_1)) * \mathsf{iam}(t)\rangle$

see below

let y = FAI(ref **this**.ticket, b) **in**

$[\mathrm{region}(S_y, \mathsf{T_{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, b, r), r) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t)]$

By the ticket permissions, as we have $(\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))} [\mathrm{T}_t]^r_1)$ and $[\mathrm{T}_t]^r_1$, we know $t \in \hat{T}$ and $t \notin_\ell T$.

(a) case non-flushed:

...

$\langle \triangleright(|\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * (\exists t' : \mathrm{TId}.\ t' \notin_\ell T * (\ulcorner \mathbf{this}.\mathsf{serving} \mapsto (n-1)\%b \urcorner * \ulcorner R \textbf{ in } t' \urcorner)\ \mathcal{U}_{t'}\ \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner) * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in \mathrm{TId}} [\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T@[t])))} [\mathrm{T}_t]^r_1) * \circledast_{n+m \le i < n+b} (\circledast_{t \in \mathrm{TId}} [\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+b} (\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}} [\mathrm{L}^t_{i\%b}]^r_1)) * \mathsf{iam}(t)\rangle$

$\langle \triangleright(|\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * t' \notin_\ell T * (\ulcorner \mathbf{this}.\mathsf{serving} \mapsto (n-1)\%b \urcorner * \ulcorner R \textbf{ in } t' \urcorner)\ \mathcal{U}_{t'}\ \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in \mathrm{TId}} [\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T@[t])))} [\mathrm{T}_t]^r_1) * \circledast_{n+m \le i < n+b} (\circledast_{t \in \mathrm{TId}} [\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+b} (\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}} [\mathrm{L}^t_{i\%b}]^r_1)) * \mathsf{iam}(t)\rangle$

see below

i. case $t = t'$:

$\langle \triangleright(|\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * t \notin_\ell T * (\ulcorner \mathbf{this}.\mathsf{serving} \mapsto (n-1)\%b \urcorner * \ulcorner R \textbf{ in } t \urcorner)\ \mathcal{U}_{t}\ \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in \mathrm{TId}} [\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T@[t])))} [\mathrm{T}_t]^r_1) * \circledast_{n+m \le i < n+b} (\circledast_{t \in \mathrm{TId}} [\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+b} (\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}} [\mathrm{L}^t_{i\%b}]^r_1)) * \mathsf{iam}(t)\rangle$

let y = FAI(ref **this**.ticket, b) **in**

$\langle \mathsf{y} = (n+m)\%b * |\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * t \notin_\ell T * \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m+1)\%b \urcorner * (\circledast_{t \in \mathrm{TId}} [\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T@[t])))} [\mathrm{T}_t]^r_1) * \circledast_{n+m \le i < n+b} (\circledast_{t \in \mathrm{TId}} [\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+b} (\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}} [\mathrm{L}^t_{i\%b}]^r_1) * \mathsf{iam}(t)\rangle$

$\langle \mathsf{y} = (n+m)\%b * |\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * t \notin_\ell T * \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m+1)\%b \urcorner * (\circledast_{t \in \mathrm{TId}} [\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T@[t])))} [\mathrm{T}_t]^r_1) * \circledast_{n+m+1 \le i < n+b} (\circledast_{t \in \mathrm{TId}} [\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+1+b} (\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}} [\mathrm{L}^t_{i\%b}]^r_1) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t)\rangle$

$\langle \mathsf{y} = (n+m)\%b * |\hat{T}| = b * |T@[t]|_\ell = m+1 * \mathrm{norepeat}(T@[t]) * \mathrm{ran}(T@[t]) \subseteq \hat{T} * (\ldots \vee \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner) * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m+1)\%b \urcorner * (\circledast_{t \in \mathrm{TId}} [\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T@[t])))} [\mathrm{T}_t]^r_1) * \circledast_{n+m+1 \le i < n+b} (\circledast_{t \in \mathrm{TId}} [\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+1+b} (\circledast_{t \in \mathrm{TId} \setminus \{(T@[t])[i-(n+b)]\}} [\mathrm{L}^t_{i\%b}]^r_1) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t)\rangle$

$\langle \mathsf{y} = (n+m)\%b * I_{\mathsf{lock}}(\mathbf{this}, R, b, r)(\mathsf{U}_{n,m+1}) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t)\rangle$

$\langle (\exists i.\ 0 \le i < m+1 \wedge (n+i)\%b = \mathsf{y}) * I_{\mathsf{lock}}(\mathbf{this}, R, b, r)(\mathsf{U}_{n,m+1}) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t)\rangle$

$[\mathrm{region}(S_{\mathsf{y}}, \mathsf{T_{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, b, r), r) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t)]$

The constraint on S_{y} holds by taking $i = m < m+1$. T can be extended to $T@[t]$ as we have $t \in \hat{T} \setminus \mathrm{ran}(T)$.

ii. case $t' \neq t$:

$\langle \ldots \rangle$

let y = FAI(ref **this**.ticket, b) **in**

$\langle \mathsf{y} = (n+m)\%b * |\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * t' \notin_\ell T * (\ulcorner \mathbf{this}.\mathsf{serving} \mapsto (n-1)\%b \urcorner * \ulcorner R \textbf{ in } t' \urcorner)\ \mathcal{U}_{t'}\ \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m+1)\%b \urcorner * (\circledast_{t \in \mathrm{TId}} [\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T@[t])))} [\mathrm{T}_t]^r_1) * \circledast_{n+m \le i < n+b} (\circledast_{t \in \mathrm{TId}} [\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+b} (\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}} [\mathrm{L}^t_{i\%b}]^r_1) * \mathsf{iam}(t)\rangle$

$\langle \mathsf{y} = (n+m)\%b * |\hat{T}| = b * |T@[t]|_\ell = m+1 * \mathrm{norepeat}(T@[t]) * \mathrm{ran}(T@[t]) \subseteq \hat{T} * t' \notin (T@[t]) * (\ulcorner \mathbf{this}.\mathsf{serving} \mapsto (n-1)\%b \urcorner * \ulcorner R \textbf{ in } t' \urcorner)\ \mathcal{U}_{t'}\ \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m+1)\%b \urcorner * (\circledast_{t \in \mathrm{TId}} [\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T@[t])))} [\mathrm{T}_t]^r_1) * \circledast_{n+m \le i < n+b} (\circledast_{t \in \mathrm{TId}} [\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+b} (\circledast_{t \in \mathrm{TId} \setminus \{(T@[t])[i-(n+b)]\}} [\mathrm{L}^t_{i\%b}]^r_1) * \mathsf{iam}(t)\rangle$

$\langle \mathsf{y} = (n+m)\%b * |\hat{T}| = b * |T@[t]|_\ell = m+1 * \mathrm{norepeat}(T@[t]) * \mathrm{ran}(T@[t]) \subseteq \hat{T} * t' \notin (T@[t]) * (\ulcorner \mathbf{this}.\mathsf{serving} \mapsto (n-1)\%b \urcorner * \ulcorner R \textbf{ in } t' \urcorner)\ \mathcal{U}_{t'}\ \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m+1)\%b \urcorner * (\circledast_{t \in \mathrm{TId}} [\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T@[t])))} [\mathrm{T}_t]^r_1) * \circledast_{n+m+1 \le i < n+b} (\circledast_{t \in \mathrm{TId}} [\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+1+b} (\circledast_{t \in \mathrm{TId} \setminus \{(T@[t])[i-(n+b)]\}} [\mathrm{L}^t_{i\%b}]^r_1) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t)\rangle$

$\langle \mathsf{y} = (n+m)\%b * I_{\mathsf{lock}}(\mathbf{this}, R, b, r)(\mathsf{U}_{n,m+1}) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t)\rangle$

$\langle (\exists i.\ 0 \le i < m+1 \wedge (n+i)\%b = \mathsf{y}) * I_{\mathsf{lock}}(\mathbf{this}, R, b, r)(\mathsf{U}_{n,m+1}) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t)\rangle$

$[\mathrm{region}(S_{\mathsf{y}}, \mathsf{T_{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, b, r), r) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t)]$

We have $t \in \hat{T} \setminus \mathrm{ran}(T)$, so $t' \notin (T@[t])$, as $t' \notin_\ell T$ and $t' \neq t$. The constraint on S_{y} holds by taking $i = m < m+1$.

(b) case flushed:

$\dots$

$\langle \triangleright(|\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in \mathrm{TId}}[\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T@[t])))}[\mathrm{T}_t]^r_1) * \circledast_{n+m \le i < n+b}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}}[\mathrm{L}^t_{i\%b}]^r_1)) * \mathsf{iam}(t) \rangle$

let $\mathsf{y} = \mathsf{FAI}(\mathsf{ref}\ \mathbf{this}.\mathsf{ticket},\ \mathsf{b})$ **in**

$\langle \mathsf{y} = (n+m)\%b * |\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m+1)\%b \urcorner * (\circledast_{t \in \mathrm{TId}}[\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T@[t])))}[\mathrm{T}_t]^r_1) * \circledast_{n+m \le i < n+b}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}}[\mathrm{L}^t_{i\%b}]^r_1) * \mathsf{iam}(t) \rangle$

$\langle \mathsf{y} = (n+m)\%b * |\hat{T}| = b * |T@[t]|_\ell = m+1 * \mathrm{norepeat}(T@[t]) * \mathrm{ran}(T@[t]) \subseteq \hat{T} * (\dots \vee \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner) * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m+1)\%b \urcorner * (\circledast_{t \in \mathrm{TId}}[\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T@[t])))}[\mathrm{T}_t]^r_1) * \circledast_{n+m+1 \le i < n+b}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+1+b}(\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}}[\mathrm{L}^t_{i\%b}]^r_1) * \mathsf{iam}(t) \rangle$

$\langle \mathsf{y} = (n+m)\%b * I_{\mathsf{lock}}(\mathbf{this}, R, b, r)(\mathsf{U}_{n,m+1}) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t) \rangle$

$\langle (\exists i.\ 0 \le i < m+1 \wedge (n+i)\%b = \mathsf{y}) * I_{\mathsf{lock}}(\mathbf{this}, R, b, r)(\mathsf{U}_{n,m+1}) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t) \rangle$

$[\mathsf{region}(S_{\mathsf{y}}, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, b, r), r) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t)]$

T can be extended to $T@[t]$ as we have $t \in \hat{T} \setminus \mathrm{ran}(T)$. The constraint on S_{y} holds by taking $i = m < m+1$.

2. case L:

$[\mathsf{region}(U \cup L, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, b, r), r) * [\mathrm{T}_t]^r_1 * \mathsf{iam}(t)]$

$\langle \triangleright I_{\mathsf{lock}}(\mathbf{this}, R, b, r)(\mathsf{L}_{n,m}) * [\mathrm{T}_t]^r_1 * \mathsf{iam}(t) \rangle$

$\langle \triangleright(\exists \hat{T} : \mathrm{set}\ \mathrm{TId}.\ \exists T : \mathrm{seq}\ \mathrm{TId}.\ |\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))}[\mathrm{T}_t]^r_1) * (\circledast_{t \in \mathrm{TId} \setminus \{T[0]\}}[\mathrm{U}_t]^t_1) * \circledast_{n+m \le i < n+b+1}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b+1 \le i < n+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{\mathrm{tl}(T)[i-(n+b+1)]\}}[\mathrm{L}^t_{i\%b}]^r_1)) * [\mathrm{T}_t]^r_1 * \mathsf{iam}(t) \rangle$

$\langle \triangleright(|\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T@[t])))}[\mathrm{T}_t]^r_1) * (\circledast_{t \in \mathrm{TId} \setminus \{T[0]\}}[\mathrm{U}_t]^t_1) * \circledast_{n+m \le i < n+b+1}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b+1 \le i < n+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{\mathrm{tl}(T)[i-(n+b+1)]\}}[\mathrm{L}^t_{i\%b}]^r_1)) * \mathsf{iam}(t) \rangle$

let $\mathsf{y} = \mathsf{FAI}(\mathsf{ref}\ \mathbf{this}.\mathsf{ticket},\ \mathsf{b})$ **in**

$\langle \mathsf{y} = (n+m)\%b * |\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T@[t])))}[\mathrm{T}_t]^r_1) * (\circledast_{t \in \mathrm{TId} \setminus \{T[0]\}}[\mathrm{U}_t]^t_1) * \circledast_{n+m \le i < n+b+1}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b+1 \le i < n+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{\mathrm{tl}(T)[i-(n+b+1)]\}}[\mathrm{L}^t_{i\%b}]^r_1) * \mathsf{iam}(t) \rangle$

$\langle \mathsf{y} = (n+m)\%b * |\hat{T}| = b * |T@[t]|_\ell = m+1 * \mathrm{norepeat}(T@[t]) * \mathrm{ran}(T@[t]) \subseteq \hat{T} * \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T@[t])))}[\mathrm{T}_t]^r_1) * (\circledast_{t \in \mathrm{TId} \setminus \{T[0]\}}[\mathrm{U}_t]^t_1) * \circledast_{n+m+1 \le i < n+b+1}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b+1 \le i < n+m+1+b}(\circledast_{t \in \mathrm{TId} \setminus \{\mathrm{tl}(T@[t])[i-(n+b+1)]\}}[\mathrm{L}^t_{i\%b}]^r_1) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t) \rangle$

$\langle \mathsf{y} = (n+m)\%b * I_{\mathsf{lock}}(\mathbf{this}, R, b, r)(\mathsf{L}_{n,m+1}) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t) \rangle$

$\langle (\exists i.\ 0 < i < m+1 \wedge (n+i)\%b = \mathsf{y}) * I_{\mathsf{lock}}(\mathbf{this}, R, b, r)(\mathsf{L}_{n,m+1}) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t) \rangle$

$[\mathsf{region}(S_{\mathsf{y}}, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, b, r), r) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t)]$

Note that $m+1 \le b$ by permissions. The constraint on S_{y} holds by taking $i = m < m+1$.

8.6.2 lock

1. case U:

$[\mathsf{region}(S_{\mathsf{y}}, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, b, r), r) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t)]$

$\langle \triangleright(I_{\mathsf{lock}}(\mathbf{this}, R, b, r)(\mathsf{U}_{n,m})) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * (\exists i.\ 0 \le i < m \wedge (n+i)\%b = \mathsf{y}) * \mathsf{iam}(t) \rangle$

$\langle \triangleright(\exists \hat{T} : \mathrm{set}\ \mathrm{TId}.\ \exists T : \mathrm{seq}\ \mathrm{TId}.\ |\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * ((\exists t' : \mathrm{TId}.\ t' \notin_\ell T * (\!(\ulcorner \mathbf{this}.\mathsf{serving} \mapsto (n-1)\%b \urcorner * \ulcorner R\ \mathbf{in}\ t' \urcorner) \,\mathcal{U}_{t'}\, \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner)\!) \vee \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner) * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in \mathrm{TId}}[\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))}[\mathrm{T}_t]^r_1) * \circledast_{n+m \le i < n+b}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}}[\mathrm{L}^t_{i\%b}]^r_1)) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * (\exists i.\ 0 \le i < m \wedge (n+i)\%b = \mathsf{y}) * \mathsf{iam}(t) \rangle$

see ticket

$\langle \triangleright(|\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * ((\exists t' : \mathrm{TId}.\ t' \notin_\ell T * (\ulcorner \mathbf{this}.\mathsf{serving} \mapsto (n-1)\%b \urcorner * \ulcorner R\ \mathbf{in}\ t' \urcorner) \,\mathcal{U}_{t'}\, \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner) \vee \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner) * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in \mathrm{TId}}[\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))}[\mathrm{T}_t]^r_1) * \circledast_{n+m \le i < n+b}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}}[\mathrm{L}^t_{i\%b}]^r_1)) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * (\exists i.\ 0 \le i < m \wedge (n+i)\%b = \mathsf{y}) * \mathsf{iam}(t) \rangle$

see below

let $\mathsf{s} = \mathbf{this}.\mathsf{serving}$ **in**

$[\text{if } \mathsf{s} = \mathsf{y} \text{ then } (\mathsf{region}(L, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, b, r), r) * (\circledast_{t \in \mathrm{TId}}[\mathrm{U}_t]^r_1) * \ulcorner R \urcorner) \text{ else } (\mathsf{region}(S_{\mathsf{y}}, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, b, r), r) * [\mathrm{L}^t_{\mathsf{y}}]^r_1) * \mathsf{iam}(t)]$

(a) case non-flushed:

$\langle \triangleright(|\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * \exists t' : \mathrm{TId}.\ t' \notin_\ell T * (\ulcorner \mathbf{this}.\mathsf{serving} \mapsto (n-1)\%b \urcorner * \ulcorner R\ \mathbf{in}\ t' \urcorner) \,\mathcal{U}_{t'}\, \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in \mathrm{TId}}[\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))}[\mathrm{T}_t]^r_1) * \circledast_{n+m \le i < n+b}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}}[\mathrm{L}^t_{i\%b}]^r_1)) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * (\exists i.\ 0 \le i < m \wedge (n+i)\%b = \mathsf{y}) * \mathsf{iam}(t) \rangle$

$\langle \triangleright(|\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * t' \notin_\ell T * (\ulcorner \mathbf{this}.\mathsf{serving} \mapsto (n-1)\%b \urcorner * \ulcorner R\ \mathbf{in}\ t' \urcorner) \,\mathcal{U}_{t'}\, \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in \mathrm{TId}}[\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))}[\mathrm{T}_t]^r_1) * \circledast_{n+m \le i < n+b}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}}[\mathrm{L}^t_{i\%b}]^r_1)) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * (\exists i.\ 0 \le i < m \wedge (n+i)\%b = \mathsf{y}) * \mathsf{iam}(t) \rangle$

We can deduce $t \neq t'$ by the lock permissions: the thread has a $[\mathrm{L}^t_{\mathsf{y}}]^r_1$, and the region holds all the $[\mathrm{L}^-_-]^-_1$ except those that are in T, so $t \in T$, but $t' \notin_\ell T$.

$\langle \triangleright(|\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * t' \notin_\ell T * (\ulcorner \mathbf{this}.\mathsf{serving} \mapsto (n-1)\%b \urcorner * \ulcorner R\ \mathbf{in}\ t' \urcorner) \,\mathcal{U}_{t'}\, \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in \mathrm{TId}}[\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))}[\mathrm{T}_t]^r_1) * \circledast_{n+m \le i < n+b}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}}[\mathrm{L}^t_{i\%b}]^r_1)) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * (\exists i.\ 0 \le i < m \wedge (n+i)\%b = \mathsf{y}) * \mathsf{iam}(t) \rangle$

let $\mathsf{s} = \mathbf{this}.\mathsf{serving}$ **in**

$\langle \mathsf{s} = (n-1)\%b * (|\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * t' \notin_\ell T * (\ulcorner \mathbf{this}.\mathsf{serving} \mapsto (n-1)\%b \urcorner * \ulcorner R\ \mathbf{in}\ t' \urcorner) \,\mathcal{U}_{t'}\, \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in \mathrm{TId}}[\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))}[\mathrm{T}_t]^r_1) * \circledast_{n+m \le i < n+b}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \le i < n+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}}[\mathrm{L}^t_{i\%b}]^r_1)) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * (\exists i.\ 0 \le i < m \wedge (n+i)\%b = \mathsf{y}) * \mathsf{iam}(t) \rangle$

$\langle \mathsf{s} \neq \mathsf{y} * I_{\mathsf{lock}}(\mathbf{this}, R, b, r)(\mathsf{U}_{n,m}) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * (\exists i.\ 0 \le i < m \wedge (n+i)\%b = \mathsf{y}) * \mathsf{iam}(t) \rangle$

$[\text{if } \mathsf{s} = \mathsf{y} \text{ then } \dots \text{ else } (\mathsf{region}(S_{\mathsf{y}}, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, b, r), r) * [\mathrm{L}^t_{\mathsf{y}}]^r_1) * \mathsf{iam}(t)]$

We have $\mathsf{s} \neq \mathsf{y}$, as if $\mathsf{s} = \mathsf{y}$, then $i = m - 1$ and $m = b$, so $|T| = b$, and $t' \in \hat{T} \setminus_{s/\ell} T$, but, as T doesn't contain duplicates, and T's elements are in $\hat{T}$, $\hat{T} \setminus_{s/\ell} T$ is empty.

(b) case flushed:

$\langle \triangleright(|\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in \mathrm{TId}}[\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))}[\mathrm{T}_t]^r_1) * \circledast_{n+m \leq i < n+b}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \leq i < n+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}}[\mathrm{L}^t_{i\%b}]^r_1)) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * (\exists i.\ 0 \leq i < m \wedge (n+i)\%b = \mathsf{y}) * \mathsf{iam}(t) \rangle$

let $\mathsf{s} = \mathbf{this}.\mathsf{serving}$ **in**

$\langle \mathsf{s} = n * |\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in \mathrm{TId}}[\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))}[\mathrm{T}_t]^r_1) * \circledast_{n+m \leq i < n+b}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \leq i < n+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}}[\mathrm{L}^t_{i\%b}]^r_1) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * (\exists i.\ 0 \leq i < m \wedge (n+i)\%b = \mathsf{y}) * \mathsf{iam}(t) \rangle$

see below

$[\text{if } \mathsf{s} = \mathsf{y} \text{ then } (\mathsf{region}(L, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, b, r), r) * [\mathrm{U}_t]^r_1) \text{ else } (\mathsf{region}(S_{\mathsf{y}}, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, b, r), r) * [\mathrm{L}^t_{\mathsf{y}}]^r_1) * \mathsf{iam}(t)]$

i. case $\mathsf{s} = \mathsf{y}$:

$\langle \mathsf{s} = \mathsf{y} * \mathsf{s} = n * |\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * t' \notin_\ell T * \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in \mathrm{TId}}[\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))}[\mathrm{T}_t]^r_1) * \circledast_{n+m \leq i < n+b}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \leq i < n+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}}[\mathrm{L}^t_{i\%b}]^r_1) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * (\exists i.\ 0 \leq i < m \wedge (n+i)\%b = \mathsf{y}) * \mathsf{iam}(t) \rangle$

$\langle \mathsf{s} = \mathsf{y} * \mathsf{s} = n * |\mathrm{tl}(T)| = m - 1 * t' \notin_\ell T * \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))}[\mathrm{T}_t]^r_1) * \circledast_{n+m \leq i < n+b+1}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b+1 \leq i < n+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{\mathrm{tl} T[i-(n+b+1)]\}}[\mathrm{L}^t_{i\%b}]^r_1) * (\exists i.\ 0 \leq i < m \wedge (n+i)\%b = \mathsf{y}) * (\circledast_{t \in \mathrm{TId}}[\mathrm{U}_t]^r_1) * \ulcorner R \urcorner * \mathsf{iam}(t) \rangle$

$\langle \mathsf{s} = \mathsf{y} * \mathsf{s} = n * |\mathrm{tl}(T)| = m - 1 * t' \notin_\ell T * \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))}[\mathrm{T}_t]^r_1) * \circledast_{n+m \leq i < n+b+1}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b+1 \leq i < n+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{\mathrm{tl} T[i-(n+b+1)]\}}[\mathrm{L}^t_{i\%b}]^r_1) * (\exists i.\ 0 \leq i < m \wedge (n+i)\%b = \mathsf{y}) * (\circledast_{t \in \mathrm{TId}}[\mathrm{U}_t]^r_1) * [\mathrm{U}_t]^r_1 * \ulcorner R \urcorner * \mathsf{iam}(t) \rangle$

$\langle \mathsf{s} = \mathsf{y} * \mathsf{s} = n * |\mathrm{tl}(T)| = m - 1 * t' \notin_\ell T * \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))}[\mathrm{T}_t]^r_1) * \circledast_{n+m \leq i < n+b+1}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b+1 \leq i < n+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{\mathrm{tl} T[i-(n+b+1)]\}}[\mathrm{L}^t_{i\%b}]^r_1) * (\exists i.\ 0 \leq i < m \wedge (n+i)\%b = \mathsf{y}) * (\circledast_{t \in \mathrm{TId} \setminus \{T[0]\}}[\mathrm{U}_t]^r_1) * [\mathrm{U}_t]^r_1 * \ulcorner R \urcorner * \mathsf{iam}(t) \rangle$

$\langle \mathsf{s} = \mathsf{y} * I_{\mathsf{lock}}(\mathbf{this}, R, b, r)(\mathsf{L}_{n,m}) * [\mathrm{U}_t]^r_1 * \ulcorner R \urcorner * \mathsf{iam}(t) \rangle$

$[\text{if } \mathsf{s} = \mathsf{y} \text{ then } (\mathsf{region}(L, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, b, r), r) * [\mathrm{U}_t]^r_1 * \ulcorner R \urcorner) \text{ else } \ldots * \mathsf{iam}(t)]$

ii. case $\mathsf{s} \neq \mathsf{y}$:

$\langle \mathsf{s} \neq \mathsf{y} * \mathsf{s} = n * |\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * t' \notin_\ell T * \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n * R \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in \mathrm{TId}}[\mathrm{U}_t]^r_1) * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))}[\mathrm{T}_t]^r_1) * \circledast_{n+m \leq i < n+b}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b \leq i < n+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{T[i-(n+b)]\}}[\mathrm{L}^t_{i\%b}]^r_1) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * (\exists i.\ 0 \leq i < m \wedge (n+i)\%b = \mathsf{y}) * \mathsf{iam}(t) \rangle$

$\langle \mathsf{s} \neq \mathsf{y} * I_{\mathsf{lock}}(\mathbf{this}, R, b, r)(\mathsf{U}_{n,m}) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * (\exists i.\ 0 \leq i < m \wedge (n+i)\%b = \mathsf{y}) * \mathsf{iam}(t) \rangle$

$[\text{if } \mathsf{s} = \mathsf{y} \text{ then } \ldots \text{ else } (\mathsf{region}(S_{\mathsf{y}}, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, b, r), r) * [\mathrm{L}^t_{\mathsf{y}}]^r_1) * \mathsf{iam}(t)]$

2. case L:

$[\mathsf{region}(S^t_{\mathsf{y}}, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, b, r), r) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t)]$

$\langle \triangleright I_{\mathsf{lock}}(\mathbf{this}, R, b, r)(\mathsf{L}_{n,m}) * (\exists i.\ 0 < i < m \wedge (n+i)\%b = \mathsf{y}) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t) \rangle$

$\langle \triangleright(\exists \hat{T} : \mathrm{set}\ \mathrm{TId}.\ \exists T : \mathrm{seq}\ \mathrm{TId}.\ |\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * \ulcorner x.\mathsf{serving} \mapsto n \urcorner * \ulcorner x.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))}[\mathrm{T}_t]^r_1) * \circledast_{n+m \leq i < n+b+1}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b+1 \leq i < n+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{\mathrm{tl}(T)[i-(n+b+1)]\}}[\mathrm{L}^t_{i\%b}]^r_1)) * (\exists i.\ 0 < i < m \wedge (n+i)\%b = \mathsf{y}) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t) \rangle$

$\langle \triangleright(|\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))}[\mathrm{T}_t]^r_1) * \circledast_{n+m \leq i < n+b+1}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b+1 \leq i < n+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{\mathrm{tl}(T)[i-(n+b+1)]\}}[\mathrm{L}^t_{i\%b}]^r_1)) * (\exists i.\ 0 < i < m \wedge (n+i)\%b = \mathsf{y}) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t) \rangle$

let $\mathsf{s} = \mathbf{this}.\mathsf{serving}$ **in**

$\langle \mathsf{s} = n * |\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))}[\mathrm{T}_t]^r_1) * \circledast_{n+m \leq i < n+b+1}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b+1 \leq i < n+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{\mathrm{tl}(T)[i-(n+b+1)]\}}[\mathrm{L}^t_{i\%b}]^r_1) * (\exists i.\ 0 < i < m \wedge (n+i)\%b = \mathsf{y}) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 \rangle$

$\langle \mathsf{s} \neq \mathsf{y} * \ldots \rangle$

$\langle \mathsf{s} \neq \mathsf{y} * I_{\mathsf{lock}}(\mathbf{this}, R, b, r)(\mathsf{L}_{n,m}) * (\exists i.\ 0 < i < m \wedge (n+i)\%b = \mathsf{y}) * [\mathrm{L}^t_{\mathsf{y}}]^r_1 * \mathsf{iam}(t) \rangle$

$[\text{if } \mathsf{s} = \mathsf{y} \text{ then } \ldots \text{ else } (\mathsf{region}(S_{\mathsf{y}}, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, b, r), r) * [\mathrm{L}^t_{\mathsf{y}}]^r_1) * \mathsf{iam}(t)]$

If $\mathsf{s} = \mathsf{y}$, then the modulo equation and the bounds on i contradict.

8.6.3 release

$[\mathsf{region}(L, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, b, r), r) * [\mathrm{U}_t]^r_1 * \overline{R} * \mathsf{iam}(t)]$

$\langle \triangleright I_{\mathsf{lock}}(\mathbf{this}, R, b, r)(\mathsf{L}_{n,m}) * [\mathrm{U}_t]^r_1 * \overline{R} * \mathsf{iam}(t) \rangle$

$\langle \triangleright(\exists \hat{T} : \mathrm{set}\ \mathrm{TId}.\ \exists T : \mathrm{seq}\ \mathrm{TId}.\ |\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))}[\mathrm{T}_t]^r_1) * (\circledast_{t \in \mathrm{TId} \setminus \{T[0]\}}[\mathrm{U}_t]^r_1) * \circledast_{n+m \leq i < n+b+1}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b+1 \leq i < n+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{\mathrm{tl}(T)[i-(n+b+1)]\}}[\mathrm{L}^t_{i\%b}]^r_1)) * [\mathrm{U}_t]^r_1 * \overline{R} * \mathsf{iam}(t) \rangle$

$\langle \triangleright(|\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * \ulcorner \mathbf{this}.\mathsf{serving} \mapsto n \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (n+m)\%b \urcorner * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))}[\mathrm{T}_t]^r_1) * (\circledast_{t \in \mathrm{TId} \setminus \{T[0]\}}[\mathrm{U}_t]^r_1) * \circledast_{n+m \leq i < n+b+1}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{n+b+1 \leq i < n+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{\mathrm{tl}(T)[i-(n+b+1)]\}}[\mathrm{L}^t_{i\%b}]^r_1)) * [\mathrm{U}_t]^r_1 * \overline{R} * \mathsf{iam}(t) \rangle$

let $\mathsf{s} = \mathbf{this}.\mathsf{serving}$ **in**

$\langle |\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * \ulcorner \mathbf{this}.\mathsf{serving} \mapsto \mathsf{s} \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (\mathsf{s}+m)\%b \urcorner * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))}[\mathrm{T}_t]^r_1) * (\circledast_{t \in \mathrm{TId} \setminus \{T[0]\}}[\mathrm{U}_t]^r_1) * \circledast_{\mathsf{s}+m \leq i < \mathsf{s}+b+1}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{\mathsf{s}+b+1 \leq i < \mathsf{s}+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{\mathrm{tl}(T)[i-(\mathsf{s}+b+1)]\}}[\mathrm{L}^t_{i\%b}]^r_1) * [\mathrm{U}_t]^r_1 * \overline{R} * \mathsf{iam}(t) \rangle$

$\langle I_{\mathsf{lock}}(\mathbf{this}, R, b, r)(\mathsf{L}_{\mathsf{s},m}) * [\mathrm{U}_t]^r_1 * \overline{R} * \mathsf{iam}(t) \rangle$

$[\mathsf{region}(L_{\mathsf{s}}, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\mathbf{this}, R, b, r), r) * [\mathrm{U}_t]^r_1 * \overline{R} * \mathsf{iam}(t)]$

$\langle \triangleright I_{\mathsf{lock}}(\mathbf{this}, R, b, r)(\mathsf{L}_{\mathsf{s},m}) * [\mathrm{U}_t]^r_1 * \overline{R} * \mathsf{iam}(t) \rangle$

$\langle \triangleright(\exists \hat{T} : \mathrm{set}\ \mathrm{TId}.\ \exists T : \mathrm{seq}\ \mathrm{TId}.\ |\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * \ulcorner \mathbf{this}.\mathsf{serving} \mapsto \mathsf{s} \urcorner * \ulcorner \mathbf{this}.\mathsf{ticket} \mapsto (\mathsf{s}+m)\%b \urcorner * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))}[\mathrm{T}_t]^r_1) * (\circledast_{t \in \mathrm{TId} \setminus \{T[0]\}}[\mathrm{U}_t]^r_1) * \circledast_{\mathsf{s}+m \leq i < \mathsf{s}+b+1}(\circledast_{t \in \mathrm{TId}}[\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{\mathsf{s}+b+1 \leq i < \mathsf{s}+m+b}(\circledast_{t \in \mathrm{TId} \setminus \{\mathrm{tl}(T)[i-(\mathsf{s}+b+1)]\}}[\mathrm{L}^t_{i\%b}]^r_1)) * [\mathrm{U}_t]^r_1 * \overline{R} * \mathsf{iam}(t) \rangle$

$\langle \triangleright(|\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * \ulcorner \textbf{this}.\mathsf{serving} \mapsto \mathsf{s} \urcorner * \ulcorner \textbf{this}.\mathsf{ticket} \mapsto (\mathsf{s}+m)\%b \urcorner * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))} [\mathrm{T}_t]^r_1) * (\circledast_{t \in \mathrm{TId} \setminus \{T[0]\}} [\mathrm{U}_t]^r_1) * \circledast_{\mathsf{s}+m \le i < \mathsf{s}+b+1} (\circledast_{t \in \mathrm{TId}} [\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{\mathsf{s}+b+1 \le i < \mathsf{s}+m+b} (\circledast_{t \in \mathrm{TId} \setminus \{\mathrm{tl}(T)[i-(\mathsf{s}+b+1)]\}} [\mathrm{L}^t_{i\%b}]^r_1)) * [\mathrm{U}_t]^r_1 * \overline{R} * \mathsf{iam}(t) \rangle$

this.serving := (s + 1) % b

$\langle |\hat{T}| = b * |T|_\ell = m * \mathrm{norepeat}(T) * \mathrm{ran}(T) \subseteq \hat{T} * ((\ulcorner \textbf{this}.\mathsf{serving} \mapsto \mathsf{s} \urcorner * \ulcorner R \textbf{ in } t \urcorner)\ \mathcal{U}_t\ \ulcorner \textbf{this}.\mathsf{serving} \mapsto (\mathsf{s}+1)\%b * R \urcorner) * \ulcorner \textbf{this}.\mathsf{ticket} \mapsto (\mathsf{s}+m)\%b \urcorner * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))} [\mathrm{T}_t]^r_1) * (\circledast_{t \in \mathrm{TId} \setminus \{T[0]\}} [\mathrm{U}_t]^r_1) * \circledast_{\mathsf{s}+m \le i < \mathsf{s}+b+1} (\circledast_{t \in \mathrm{TId}} [\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{\mathsf{s}+b+1 \le i < \mathsf{s}+m+b} (\circledast_{t \in \mathrm{TId} \setminus \{\mathrm{tl}(T)[i-(\mathsf{s}+b+1)]\}} [\mathrm{L}^t_{i\%b}]^r_1) * [\mathrm{U}_t]^r_1 * \mathsf{iam}(t) \rangle$

$\langle |\hat{T}| = b * |\mathrm{tl}(T)|_\ell = m-1 * \mathrm{norepeat}(\mathrm{tl}(T)) * \mathrm{ran}(\mathrm{tl}(T)) \subseteq \hat{T} * ((\exists t' : \mathrm{TId}.\ t' \notin_\ell T * (\!|(\ulcorner \textbf{this}.\mathsf{serving} \mapsto \mathsf{s} \urcorner * \ulcorner R \textbf{ in } t' \urcorner)\ \mathcal{U}_{t'}\ \ulcorner \textbf{this}.\mathsf{serving} \mapsto (\mathsf{s}+1)\%b * R \urcorner|\!)) \vee \ldots) * \ulcorner \textbf{this}.\mathsf{ticket} \mapsto (\mathsf{s}+1+m-1)\%b \urcorner * (\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(\mathrm{tl}(T))))} [\mathrm{T}_t]^r_1) * [\mathrm{T}_t]^r_1 * (\circledast_{t \in \mathrm{TId}} [\mathrm{U}_t]^r_1) * \circledast_{\mathsf{s}+1+m-1 \le i < \mathsf{s}+b+1} (\circledast_{t \in \mathrm{TId}} [\mathrm{L}^t_{i\%b}]^r_1) * \circledast_{\mathsf{s}+b+1 \le i < \mathsf{s}+1+m-1+b} (\circledast_{t \in \mathrm{TId} \setminus \{\mathrm{tl}(T)[i-(\mathsf{s}+b+1)]\}} [\mathrm{L}^t_{i\%b}]^r_1) * \mathsf{iam}(t) \rangle$

$\langle I_{\mathsf{lock}}(\textbf{this}, R, b, r)(\mathsf{U}_{\mathsf{s}+1, m-1}) * [\mathrm{T}_t]^r_1 * \mathsf{iam}(t) \rangle$

$[\mathsf{region}(U \cup L, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\textbf{this}, R, b, r), r) * [\mathrm{T}_t]^r_1 * \mathsf{iam}(t)]$

It is possible to get a $[\mathrm{T}_t]^r_1$ out of $(\circledast_{t \in (\mathrm{TId} \setminus (\hat{T} \setminus \mathrm{ran}(T)))} [\mathrm{T}_t]^r_1)$ as $t = \mathrm{hd}(T)$ by the unlock permissions.

8.6.4 Constructor

$[\overline{\mathsf{R}} * 0 < \mathsf{b} * |T| = \mathsf{b}]$

TicketLock(int b) {

$[\overline{\mathsf{R}} * 0 < \mathsf{b} * |T| = \mathsf{b} * \ulcorner \textbf{this}.\mathsf{bound} \mapsto \textbf{null} * \textbf{this}.\mathsf{serving} \mapsto \textbf{null} * \textbf{this}.\mathsf{ticket} \mapsto \textbf{null} \urcorner]$

$[(\!|\overline{\mathsf{R}}|\!) * 0 < \mathsf{b} * |T| = \mathsf{b} * \ulcorner \textbf{this}.\mathsf{bound} \mapsto \textbf{null} * \textbf{this}.\mathsf{serving} \mapsto \textbf{null} * \textbf{this}.\mathsf{ticket} \mapsto \textbf{null} \urcorner]$

this.bound := b;

$[(\!|\overline{\mathsf{R}}|\!) * 0 < \mathsf{b} * |T| = \mathsf{b} * (\!|\ulcorner \textbf{this}.\mathsf{bound} \mapsto \mathsf{b} \textbf{ in } t \urcorner|\!) * \ulcorner \textbf{this}.\mathsf{serving} \mapsto \textbf{null} * \textbf{this}.\mathsf{ticket} \mapsto \textbf{null} \urcorner]$

this.serving := 0;
this.ticket := 0;

$[(\!|\overline{\mathsf{R}}|\!) * 0 < \mathsf{b} * |T| = \mathsf{b} * (\!|\ulcorner \textbf{this}.\mathsf{bound} \mapsto \mathsf{b} \textbf{ in } t \urcorner|\!) * (\!|\ulcorner \textbf{this}.\mathsf{serving} \mapsto 0 \textbf{ in } t \urcorner|\!) * (\!|\ulcorner \textbf{this}.\mathsf{ticket} \mapsto 0 \textbf{ in } t \urcorner|\!)]$

fence

$[\ulcorner \mathsf{R} \urcorner * 0 < \mathsf{b} * |T| = \mathsf{b} * \ulcorner \textbf{this}.\mathsf{bound} \mapsto \mathsf{b} * \textbf{this}.\mathsf{serving} \mapsto 0 * \textbf{this}.\mathsf{ticket} \mapsto 0 \urcorner]$

$[0 < \mathsf{b} * \ulcorner \textbf{this}.\mathsf{bound} \mapsto \mathsf{b} \urcorner * \exists r : \mathsf{RId}.\ \mathsf{region}(U \cup L, \mathsf{T}_{\mathsf{lock}}, I_{\mathsf{lock}}(\textbf{this}, R, b, r), r) * \circledast_{t \in T} [\mathrm{T}_t]^r_1]$

$[\circledast_{t \in T}\, mayTicket(\textbf{this}, \mathsf{R}, t)]$

}

9 Verification of a double-checked initialisation wrapper in the TSO logic

In this section we verify a double-checked initialisation [5] (also known as double-checked locking) wrapper using the TSO logic, reusing our spin-lock from Section 6.

9.1 Specification

Double-checked initialisation is a design pattern that reduces the cost of lazy initialisation by having clients only use a lock if the object has (to their knowledge) not been initialised yet. Our specification returns to the client the object's invariant (which has to be a duplicable resource because there can be multiple accesses), and guarantees that accessing the object several times returns the same object.

$$
\begin{aligned}
&\forall P : \mathsf{Prop}_{\mathsf{SC}}, Q : \mathsf{Val} \to \mathsf{Prop}_{\mathsf{SC}}.\\
&[P]\ \mathsf{T}()\ [r.\ Q(r)] \wedge (\forall x : \mathsf{Val}.\ Q(x) \Leftrightarrow Q(x) * Q(x)) \wedge \mathsf{stable}(P) \wedge (\forall x : \mathsf{Val}.\ \mathsf{stable}(Q(x))) \Rightarrow\\
&\exists mayGet : \mathsf{Val} \to \mathsf{Prop}_{\mathsf{SC}}.\\
&\exists mayGetRes : \mathsf{Val} \times \mathsf{Val} \to \mathsf{Prop}_{\mathsf{SC}}.\\
&\qquad\quad [\mathsf{emp}]\ \mathsf{DoubleCheckInitialisedT}()\ [\mathsf{r}.\ mayGet(\mathsf{r})]\\
&\quad\ \wedge\ [P * mayGet(\mathbf{this})]\ \mathsf{DoubleCheckInitialisedT.get}()\ [r.\ r \neq \mathbf{null} * mayGetRes(\mathbf{this}, r) * Q(r)]\\
&\quad\ \wedge\ \forall \mathsf{x} : \mathsf{Val}.\ mayGet(\mathsf{x}) \Leftrightarrow mayGet(\mathsf{x}) * mayGet(\mathsf{x})\\
&\quad\ \wedge\ \forall \mathsf{x} : \mathsf{Val}.\ \mathsf{stable}(mayGet(\mathsf{x}))\\
&\quad\ \wedge\ \forall \mathsf{x}, a : \mathsf{Val}.\ mayGetRes(\mathsf{x}, a) \Leftrightarrow mayGetRes(\mathsf{x}, a) * mayGetRes(\mathsf{x}, a)\\
&\quad\ \wedge\ \forall \mathsf{x} : \mathsf{Val}.\ \forall a, b : \mathsf{Val}.\ mayGetRes(\mathsf{x}, a) * mayGetRes(\mathsf{x}, b) \Longrightarrow a = b
\end{aligned}
$$

```
class T {
  T() =
    ...
  ...
}
class DoubleCheckInitialisedT {
  Lock lock;
  T obj;
  DoubleCheckInitialisedT() =
    let l = new Lock() in
    this.lock = l;
    fence;
    this
  T get() =
    let x = this.obj in
    if x = null then
```

```
    (let l = this.lock in
     l.acquire();
     let y = this.obj in
     if y = null then
       (let z = new T() in
        this.obj := z;
        l.release();
        z)
     else
       (l.release();
        y)
  else x
}
```

9.2 Definitions

The double-checked initialisation wrapper can have two states: Null, where the object has not been initialised, and Set, where the object has been initialised, but this may have not reached main memory yet. By taking the lock, a thread obtains the knowledge that the object does not have buffered updates. In the Null state, this means the object has not been initialised, so the thread can initialise the object. In the Set state, this means that the initialisation of the object has reached main memory.

$$mayGet(x) = \exists r.\mathsf{region}(\{\text{Null}, \text{Set}\}, T_{\text{dci}}, I_{\text{dci}}(x,r), r) * \text{isLock}(x.\mathsf{lock}, \ulcorner x.\mathsf{obj} \xrightarrow{1/2} _ \urcorner * [\text{SET}]^r_1)$$
$$mayGetRes(x, v) = (\!|\overline{x.\mathsf{obj} \mapsto v}|\!)$$

Interpretation:

$$I_{\text{dci}}(x,r)(\text{Null}) = \ulcorner x.\mathsf{obj} \xrightarrow{1/2} \mathbf{null} \urcorner$$
$$I_{\text{dci}}(x,r)(\text{Set}) = \exists t : \text{TId}. \exists v : \mathsf{Val}.\ v \neq \mathbf{null} * (\!|\ulcorner x.\mathsf{obj} \xrightarrow{1/2} \mathbf{null} \urcorner\, \mathcal{U}_t\, (\ulcorner x.\mathsf{obj} \mapsto v \urcorner * Q(v))|\!)$$

Transitions of the protocol:

$$\text{SET} : \text{Null} \to \text{Set}$$

The idea is that the lock gives us $\ulcorner \mathbf{this}.\mathsf{obj} \xrightarrow{1/2} _ \urcorner$, while the region only gives us $(\!|\ulcorner \mathbf{this}.\mathsf{obj} \xrightarrow{1/2} v_1 \urcorner\, \mathcal{U}\, \ulcorner \mathbf{this}.\mathsf{obj} \mapsto v_2 \urcorner|\!)$, but as $v_1 \neq v_2$, we don't have to consider the case where there is a buffered write.

9.3 Proof outline

9.3.1 Constructor

DoubleCheckInitialisedT() {

$[\ulcorner \textbf{this}.\mathsf{obj} \mapsto \textbf{null} \urcorner * \ulcorner \textbf{this}.\mathsf{lock} \mapsto \textbf{null} \urcorner]$

$[\exists r.\ \mathsf{region}(\{\mathrm{Null}, \mathrm{Set}\}, T_{\mathrm{dci}}, I_{\mathrm{dci}}(\textbf{this}, r), r) * [\mathrm{SET}]^r_1 * \ulcorner \textbf{this}.\mathsf{obj} \xmapsto{1/2} \textbf{null} \urcorner * \ulcorner \textbf{this}.\mathsf{lock} \mapsto \textbf{null} \urcorner]$

let l = new Lock() **in**

$[\mathsf{region}(\{\mathrm{Null}, \mathrm{Set}\}, T_{\mathrm{dci}}, I_{\mathrm{dci}}(\textbf{this}, r), r) * \ulcorner \mathrm{isLock}(l, \ulcorner \textbf{this}.\mathsf{obj} \xmapsto{1/2} _ \urcorner * [\mathrm{SET}]^r_1) \urcorner * \ulcorner \textbf{this}.\mathsf{lock} \mapsto \textbf{null} \urcorner]$

this.lock = l;

$[\mathsf{region}(\{\mathrm{Null}, \mathrm{Set}\}, T_{\mathrm{dci}}, I_{\mathrm{dci}}(\textbf{this}, r), r) * \overline{\mathrm{isLock}(\textbf{this}.\mathsf{lock}, \ulcorner \textbf{this}.\mathsf{obj} \xmapsto{1/2} _ \urcorner * [\mathrm{SET}]^r_1)}]$

fence

$[\mathsf{region}(\{\mathrm{Null}, \mathrm{Set}\}, T_{\mathrm{dci}}, I_{\mathrm{dci}}(\textbf{this}, r), r) * \ulcorner \mathrm{isLock}(\textbf{this}.\mathsf{lock}, \ulcorner \textbf{this}.\mathsf{obj} \xmapsto{1/2} _ \urcorner * [\mathrm{SET}]^r_1) \urcorner]$

$[\mathit{mayGet}(\textbf{this})]$

}

9.3.2 get

get() {

$[\mathit{mayGet}(\textbf{this}) * P]$

$[\mathsf{region}(\{\mathrm{Null}, \mathrm{Set}\}, T_{\mathrm{dci}}, I_{\mathrm{dci}}(\textbf{this}, r), r) * \mathrm{isLock}(\textbf{this}.\mathsf{lock}, \ulcorner \textbf{this}.\mathsf{obj} \xmapsto{1/2} _ \urcorner * [\mathrm{SET}]^r_1)]$

let x = **this**.obj **in**

$[\exists v.\ \mathsf{x} = v * (\mathsf{x} \neq \textbf{null} \Rightarrow \mathit{mayGetRes}(\textbf{this}, \mathsf{x}) * Q(\mathsf{x})) * \mathsf{region}(\{\mathrm{Null}, \mathrm{Set}\}, T_{\mathrm{dci}}, I_{\mathrm{dci}}(\textbf{this}, r), r) * \mathrm{isLock}(\textbf{this}.\mathsf{lock}, \ulcorner \textbf{this}.\mathsf{obj} \xmapsto{1/2} _ \urcorner * [\mathrm{SET}]^r_1) * P]$

if x = null **then**

($[\mathsf{region}(\{\mathrm{Null}, \mathrm{Set}\}, T_{\mathrm{dci}}, I_{\mathrm{dci}}(\textbf{this}, r), r) * \mathrm{isLock}(\textbf{this}.\mathsf{lock}, \ulcorner \textbf{this}.\mathsf{obj} \xmapsto{1/2} _ \urcorner * [\mathrm{SET}]^r_1) * P]$

let l = **this**.lock **in**

$[\mathsf{region}(\{\mathrm{Null}, \mathrm{Set}\}, T_{\mathrm{dci}}, I_{\mathrm{dci}}(\textbf{this}, r), r) * \mathrm{isLock}(\mathsf{l}, \ulcorner \textbf{this}.\mathsf{obj} \xmapsto{1/2} _ \urcorner * [\mathrm{SET}]^r_1) * P]$

l.acquire();

$[\mathsf{region}(\{\mathrm{Null}, \mathrm{Set}\}, T_{\mathrm{dci}}, I_{\mathrm{dci}}(\textbf{this}, r), r) * \ulcorner \textbf{this}.\mathsf{obj} \xmapsto{1/2} _ \urcorner * [\mathrm{SET}]^r_1 * \mathrm{locked}(\mathsf{l}, \ulcorner \textbf{this}.\mathsf{obj} \xmapsto{1/2} _ \urcorner * [\mathrm{SET}]^r_1) * P]$

let y = **this**.obj **in**

$[(\mathsf{y} = \textbf{null} \Rightarrow \mathsf{region}(\{\mathrm{Null}\}, T_{\mathrm{dci}}, I_{\mathrm{dci}}(\textbf{this}, r), r)) * (\mathsf{y} \neq \textbf{null} \Rightarrow \mathit{mayGetRes}(\textbf{this}, \mathsf{y}) * Q(\mathsf{y})) * \ulcorner \textbf{this}.\mathsf{obj} \xmapsto{1/2} _ \urcorner * [\mathrm{SET}]^r_1 * \mathrm{locked}(\mathsf{l}, \ulcorner \textbf{this}.\mathsf{obj} \xmapsto{1/2} _ \urcorner * [\mathrm{SET}]^r_1) * P]$

if y = null **then**

$[\mathsf{region}(\{\mathrm{Null}\}, T_{\mathrm{dci}}, I_{\mathrm{dci}}(\textbf{this}, r), r) * \ulcorner \textbf{this}.\mathsf{obj} \xmapsto{1/2} _ \urcorner * [\mathrm{SET}]^r_1 * \mathrm{locked}(\mathsf{l}, \ulcorner \textbf{this}.\mathsf{obj} \xmapsto{1/2} _ \urcorner * [\mathrm{SET}]^r_1) * P]$

let z = **this**.obj **in**

$[\mathsf{z} \neq \textbf{null} * \mathsf{region}(\{\mathrm{Null}\}, T_{\mathrm{dci}}, I_{\mathrm{dci}}(\textbf{this}, r), r) * \ulcorner \textbf{this}.\mathsf{obj} \xmapsto{1/2} _ \urcorner * [\mathrm{SET}]^r_1 * \mathrm{locked}(\mathsf{l}, \ulcorner \textbf{this}.\mathsf{obj} \xmapsto{1/2} _ \urcorner * [\mathrm{SET}]^r_1) * Q(\mathsf{z})]$

this.obj = z;

$[\mathsf{z} \neq \textbf{null} * \mathsf{region}(\{\mathrm{Set}\}, T_{\mathrm{dci}}, I_{\mathrm{dci}}(\textbf{this}, r), r) * \overline{\textbf{this}.\mathsf{obj} \xmapsto{1/2} _} * [\mathrm{SET}]^r_1 * \mathrm{locked}(\mathsf{l}, \ulcorner \textbf{this}.\mathsf{obj} \xmapsto{1/2} _ \urcorner * [\mathrm{SET}]^r_1) * Q(\mathsf{z})]$

$[\mathsf{z} \neq \textbf{null} * \mathit{mayGetRes}(\textbf{this}, \mathsf{z}) * \textbf{this}.\mathsf{obj} \xmapsto{1/2} _ * [\mathrm{SET}]^r_1 * \mathrm{locked}(\mathsf{l}, \ulcorner \textbf{this}.\mathsf{obj} \xmapsto{1/2} _ \urcorner * [\mathrm{SET}]^r_1) * Q(\mathsf{z})]$

l.release();

$[\mathsf{z} \neq \textbf{null} * \mathit{mayGetRes}(\textbf{this}, \mathsf{z}) * Q(\mathsf{z})]$

z

else

$[\mathsf{y} \neq \textbf{null} * \mathit{mayGetRes}(\textbf{this}, \mathsf{y}) * Q(\mathsf{y}) * \overline{\textbf{this}.\mathsf{obj} \xmapsto{1/2} _} * [\mathrm{SET}]^r_1 * \mathrm{locked}(\mathsf{l}, \ulcorner \textbf{this}.\mathsf{obj} \xmapsto{1/2} _ \urcorner * [\mathrm{SET}]^r_1)]$

$[\mathsf{y} \neq \textbf{null} * \mathit{mayGetRes}(\textbf{this}, \mathsf{y}) * Q(\mathsf{y}) * \textbf{this}.\mathsf{obj} \xmapsto{1/2} _ * [\mathrm{SET}]^r_1 * \mathrm{locked}(\mathsf{l}, \ulcorner \textbf{this}.\mathsf{obj} \xmapsto{1/2} _ \urcorner * [\mathrm{SET}]^r_1) * Q(\mathsf{y})]$

l.release();

$[\mathsf{y} \neq \textbf{null} * \mathit{mayGetRes}(\textbf{this}, \mathsf{y}) * Q(\mathsf{y})]$

y
else
$[\mathsf{x} \neq \mathbf{null} * \mathit{mayGetRes}(\mathbf{this}, \mathsf{x}) * Q(\mathsf{x})]$
x
}

9.3.3 Duplicability

$$\mathit{mayGet}(x) \Leftrightarrow \mathit{mayGet}(x) * \mathit{mayGet}(x)$$

follows from isLock being duplicable, and region assertions being duplicable.

9.3.4 Equality of results

$$\mathit{mayGetRes}(x,a) \Leftrightarrow \mathit{mayGetRes}(x,a) * \mathit{mayGetRes}(x,a)$$

and

$$\mathit{mayGetRes}(x,a) * \mathit{mayGetRes}(x,b) = (\!|\overline{x.\mathsf{obj} \Rrightarrow a}|\!) * (\!|\overline{x.\mathsf{obj} \Rrightarrow b}|\!) \Rightarrow a = b$$

9.4 Proof detail

The interesting part of the proof is

$[\mathsf{region}(\{\mathrm{Null},\mathrm{Set}\}, T_{\mathrm{dci}}, I_{\mathrm{dci}}(\mathbf{this}, r), r) * \ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} _ \urcorner * [\mathrm{SET}]^r_1 * \mathrm{locked}(\mathsf{l}, \ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} _ \urcorner * [\mathrm{SET}]^r_1)]$

$[\mathsf{region}(\{\mathrm{Null},\mathrm{Set}\}, T_{\mathrm{dci}}, I_{\mathrm{dci}}(\mathbf{this}, r), r) * \ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} _ \urcorner * [\mathrm{SET}]^r_1]$

let y = **this**.obj **in**

$[(\mathsf{y} = \mathbf{null} \Rightarrow \mathsf{region}(\{\mathrm{Null}\}, T_{\mathrm{dci}}, I_{\mathrm{dci}}(\mathbf{this}, r), r)) * (\mathsf{y} \neq \mathbf{null} \Rightarrow \mathit{mayGetRes}(\mathbf{this}, \mathsf{y}) * Q(\mathsf{y})) * \ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} _ \urcorner * [\mathrm{SET}]^r_1]$

$[(\mathsf{y} = \mathbf{null} \Rightarrow \mathsf{region}(\{\mathrm{Null}\}, T_{\mathrm{dci}}, I_{\mathrm{dci}}(\mathbf{this}, r), r)) * (\mathsf{y} \neq \mathbf{null} \Rightarrow \mathit{mayGetRes}(\mathbf{this}, \mathsf{y}) * Q(\mathsf{y})) * \ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} _ \urcorner * [\mathrm{SET}]^r_1 * \mathrm{locked}(\mathsf{l}, \ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} _ \urcorner * [\mathrm{SET}]^r_1)]$

1. case Null:

   $[\mathsf{region}(\{\mathrm{Null}\}, T_{\mathrm{dci}}, I_{\mathrm{dci}}(\mathbf{this}, r), r) * \ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} _ \urcorner * [\mathrm{SET}]^r_1]$

   $\langle (\triangleright I_{\mathrm{dci}}(\mathbf{this}, r)(\mathrm{Null})) * \ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} _ \urcorner * [\mathrm{SET}]^r_1 \rangle$

   $\langle (\triangleright \ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} \mathbf{null} \urcorner) * \ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} _ \urcorner * [\mathrm{SET}]^r_1 \rangle$

   let y = **this**.obj **in**

   $\langle \ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} \mathbf{null} \urcorner * \ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} \mathsf{y} \urcorner * [\mathrm{SET}]^r_1 \rangle$

   $\langle \mathsf{y} = \mathbf{null} * \ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} \mathbf{null} \urcorner * \ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} \mathsf{y} \urcorner * [\mathrm{SET}]^r_1 \rangle$

   $[(\mathsf{y} = \mathbf{null} \Rightarrow \mathsf{region}(\{\mathrm{Null}\}, T_{\mathrm{dci}}, I_{\mathrm{dci}}(\mathbf{this}, r), r)) * (\mathsf{y} \neq \mathbf{null} \Rightarrow \mathit{mayGetRes}(\mathbf{this}, \mathsf{y}) * Q(\mathsf{y})) * \ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} _ \urcorner * [\mathrm{SET}]^r_1]$

2. case Set:

   $[\mathsf{region}(\{\mathrm{Set}\}, T_{\mathrm{dci}}, I_{\mathrm{dci}}(\mathbf{this}, r), r) * \ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} _ \urcorner * [\mathrm{SET}]^r_1]$

   $\langle \triangleright (I_{\mathrm{dci}}(\mathbf{this}, r)(\mathrm{Set})) * \ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} _ \urcorner * [\mathrm{SET}]^r_1 \rangle$

   $\langle \triangleright (\exists t' : \mathrm{TId}. \exists v : \mathsf{Val}.\ v \neq \mathbf{null} * (\!| \ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} \mathbf{null} \urcorner \, \mathcal{U}_{t'} \, (\ulcorner \mathbf{this}.\mathsf{obj} \Rrightarrow v \urcorner * Q(v)) |\!)) * \ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} _ \urcorner * [\mathrm{SET}]^r_1 \rangle$

   $\langle \triangleright (v \neq \mathbf{null} * (\!| \ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} \mathbf{null} \urcorner \, \mathcal{U}_{t'} \, (\ulcorner \mathbf{this}.\mathsf{obj} \Rrightarrow v \urcorner * Q(v)) |\!)) * \ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} _ \urcorner * [\mathrm{SET}]^r_1 \rangle$

   $\langle \triangleright (v \neq \mathbf{null} * (\ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} v \urcorner \vee \ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} \mathbf{null} \urcorner \, \mathcal{U}_{t'} \, (\ulcorner \mathbf{this}.\mathsf{obj} \Rrightarrow v \urcorner * Q(v)))) * \ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} _ \urcorner * [\mathrm{SET}]^r_1 \rangle$

   ⟨see below⟩

   let y = **this**.obj **in**

   $[(\mathsf{y} = \mathbf{null} \Rightarrow \mathsf{region}(\{\mathrm{Null}\}, T_{\mathrm{dci}}, I_{\mathrm{dci}}(\mathbf{this}, r), r)) * (\mathsf{y} \neq \mathbf{null} \Rightarrow \mathit{mayGetRes}(\mathbf{this}, \mathsf{y}) * Q(\mathsf{y})) * \ulcorner \mathbf{this}.\mathsf{obj} \overset{1/2}{\longmapsto} _ \urcorner * [\mathrm{SET}]^r_1]$

(a) case left:

$\langle \triangleright(v \neq \textsf{null} * \ulcorner \textsf{this.obj} \xmapsto{1/2} v \urcorner) * \ulcorner \textsf{this.obj} \xmapsto{1/2} _ \urcorner * [\textsc{Set}]^r_1 \rangle$
let y = **this**.obj **in**
$\langle \textsf{y} \neq \textsf{null} * \ulcorner \textsf{this.obj} \Mapsto \textsf{y} \urcorner * \ulcorner \textsf{this.obj} \Mapsto \textsf{y} \urcorner * \ulcorner \textsf{this.obj} \xmapsto{1/2} _ \urcorner * [\textsc{Set}]^r_1 \rangle$
$[(\textsf{y} = \textsf{null} \Rightarrow \textsf{region}(\{\text{Null}\}, T_{\text{dci}}, I_{\text{dci}}(\textsf{this}, r), r)) * (\textsf{y} \neq \textsf{null} \Rightarrow \mathit{mayGetRes}(\textsf{this}, \textsf{y}) * Q(\textsf{y})) * \ulcorner \textsf{this.obj} \xmapsto{1/2} _ \urcorner * [\textsc{Set}]^r_1]$

(b) case right:

$\langle \triangleright(v \neq \textsf{null} * (\ulcorner \textsf{this.obj} \xmapsto{1/2} \textsf{null} \urcorner \, \mathcal{U}_{t'} \, (\ulcorner \textsf{this.obj} \Mapsto v \urcorner * Q(v)))) * \ulcorner \textsf{this.obj} \xmapsto{1/2} _ \urcorner * [\textsc{Set}]^r_1 \rangle$
$\langle \triangleright(v \neq \textsf{null} * (\ulcorner \textsf{this.obj} \xmapsto{1/2} \textsf{null} \urcorner \, \mathcal{U}_{t'} \, (\ulcorner \textsf{this.obj} \Mapsto v \urcorner * Q(v))) * \ulcorner \textsf{this.obj} \xmapsto{1/2} _ \urcorner) * [\textsc{Set}]^r_1 \rangle$
$\langle \triangleright(\textsf{false}) * [\textsc{Set}]^r_1 \rangle$
let y = **this**.obj **in**
$[(\textsf{y} = \textsf{null} \Rightarrow \textsf{region}(\{\text{Null}\}, T_{\text{dci}}, I_{\text{dci}}(\textsf{this}, r), r)) * (\textsf{y} \neq \textsf{null} \Rightarrow \mathit{mayGetRes}(\textsf{this}, \textsf{y}) * Q(\textsf{y})) * \ulcorner \textsf{this.obj} \xmapsto{1/2} _ \urcorner * [\textsc{Set}]^r_1]$

This is where we use the trick of combining the content of the region with the content of the lock to derive a contradiction: We have

$$v \neq \textsf{null} * \ulcorner \textsf{this.obj} \xmapsto{1/2} \textsf{null} \urcorner \, \mathcal{U}_{t'} \, (\ulcorner \textsf{this.obj} \Mapsto v \urcorner * Q(v)) * \ulcorner \textsf{this.obj} \xmapsto{1/2} _ \urcorner$$

We can rewrite $\ulcorner \textsf{this.obj} \xmapsto{1/2} _ \urcorner$ as $\exists v'.\ \ulcorner \textsf{this.obj} \xmapsto{1/2} v' \urcorner$, and push it inside the $\mathcal{U}$ using

$$(P \, \mathcal{U} \, Q) * \ulcorner R \urcorner \vdash (P * \ulcorner R \urcorner) \, \mathcal{U} \, (Q * \ulcorner R \urcorner)$$

to get

$$\exists v'.\ v \neq \textsf{null} * (\ulcorner \textsf{this.obj} \xmapsto{1/2} \textsf{null} * \textsf{this.obj} \xmapsto{1/2} v' \urcorner \, \mathcal{U}_{t'} \, (\ulcorner \textsf{this.obj} \Mapsto v * \textsf{this.obj} \xmapsto{1/2} v' \urcorner * Q(v)))$$

Now we perform a case analysis on $v' = \textsf{null}$. If $v' = \textsf{null}$, then the right-hand side of the $\mathcal{U}$ implies $v = v'$, which implies $\textsf{null} \neq \textsf{null}$, which is **false**, which means (as $P \, \mathcal{U} \, \textsf{false} \vdash \textsf{false}$) that the whole assertion is **false**. If $v' \neq \textsf{null}$, then the left-hand side of the $\mathcal{U}$ implies $v' = \textsf{null}$, which is **false**, which means (as $\textsf{false} \, \mathcal{U} \, Q \vdash \textsf{false}$ that the whole assertion is **false**).

10 Verification of a circular buffer in an extension of the TSO logic

In this section we verify a circular buffer [3] implementation using the TSO logic extended with arrays.

10.1 Treatment of arrays

Although the model and the logic don't include arrays, we can extended the syntax with array creation $\mathsf{x.f} := \textbf{new}\ \mathsf{ty[N]}$, array access $\mathsf{x.f[i]}$, and array writing $\mathsf{x.f[i]} := \mathsf{e}$, and add the following axioms:

$$[0 < \mathsf{N} * \ulcorner \mathsf{x.f} \mapsto \textbf{null} \urcorner]\ \mathsf{x.f} := \textbf{new}\ \mathsf{ty[N]}\ [\ulcorner \mathsf{x.f[0]} \mapsto \textbf{null} * \ldots \mathsf{x.f[N-1]} \mapsto \textbf{null} \urcorner]$$

and

$$[\ulcorner \mathsf{x.f[i]} \mapsto \mathsf{v1} \urcorner]\ \mathsf{x.f[i]} := v2\ [\ulcorner \mathsf{x.f[i]} \mapsto \mathsf{v1} \urcorner\, \mathcal{U}\, \ulcorner \mathsf{x.f[i]} \mapsto \mathsf{v2} \urcorner]$$

10.2 Specification

A circular buffer is a single-writer single-reader resource transfer mechanism.
Our specification allows the writer to transfer resources to the reader. Note that as for the ticket lock (see section 8), the reader and writer permissions have to be attached to threads.
This algorithm is interesting in TSO because it does not need any synchronisation operations: the FIFO nature of TSO buffers is enough.

$$\begin{array}{l}
\exists Prod, Cons : \mathsf{Val} \times \mathsf{Prop_{sc}} \to \mathsf{Prop_{sc}}.\\
\quad \forall P : \mathsf{Val} \to \mathsf{Prop_{sc}}.\ \mathsf{stable(R)} \ \Rightarrow\\
\qquad\qquad \forall tr, tw : \mathrm{TId}, \mathsf{N} : \mathbb{N}.\ [tw \neq tr * 0 < \mathsf{N}]\ \mathsf{CircularBuffer(N)}\ [r.\ Prod(r, P, tw) * Cons(r, P, tr)]\\
\qquad \wedge\ \ \forall \mathsf{x} : \mathsf{Val}.\ [Prod(\textbf{this}, P, tw) * \overline{P(\mathsf{x})} * \mathsf{iam}(tw)]\ \mathsf{CircularBuffer.put(x)}\ [Prod(\textbf{this}, P, tw)]\\
\qquad \wedge\ \ [Cons(\textbf{this}, \mathsf{R}, tr) * \mathsf{iam}(tr)]\ \mathsf{CircularBuffer.get()}\ [r.\ Cons(\textbf{this}, P, tr) * \ulcorner P(r) \urcorner]\\
\qquad \wedge\ \ \forall \mathsf{x} : \mathsf{Val}.\forall t : \mathrm{TId}.\ (\mathsf{stable}(Prod(\mathsf{x}, P, t)) \wedge \mathsf{stable}(Cons(\mathsf{x}, P, t)))
\end{array}$$

```
class CircularBuffer {
  int[] buf;
  int wi;
  int ri;
  int N;
  CircularBuffer() =
    (this.buf := new int[N];
     this.N := N;
     this.ri := 0;
```

```
    this.wi := 0;
    fence;
    this)
  void put(int v) =
    let N = this.N in
    let w = this.wi in
    let r = this.ri in
    let w' = (w + 1) % N in
    if w' == r then put(v)
    else
      (this.buf[w] := v;
       this.wi := w')
  int get() =
    let N = this.N in
    let r = this.ri in
    let w = this.wi in
    let r' = (r + 1) % N in
    if w == r' then get()
    else
      (let v = this.buf[r] in
       this.ri := r';
       v)
}
```

10.3 Definitions

The circular buffer relies on three properties:

1. As the circular buffer does not include any synchronisation operations, the apparent state of both the reader and the writer can lag behind by several slots. However, because TSO buffers are FIFO, once a write reaches main memory, all the previous ones have also reached main memory. Therefore, the reader's knowledge of the state of the writer can only go forward.
2. The writer can't overtake the reader index by the invariant of the circular buffer.
3. Finally, the reader has the exclusive right to change the reader index.

This means that after seeing that the writer index is ahead of the reader index, the reader is sure to find a resource available, because if the increment to the writer index has propagated to main memory, then so has the resource that it advertises. The writer works symmetrically.

The first property is encoded in the states of the region: when it reads a certain value for the writer index, the reader *moves* the region to a state where the writer index is at least that value. This move does not correspond to a physical change, but to a

logical change: the reader *realises* that the region is in a certain physical state. The stabilisation closure of this state is the set of states with greater writer index and (by the third property) the same reader index, but where the writer index hasn't overtaken the reader index (by the second property). Again, the writer works symmetrically.

Notation To reduce clutter when writing the set of states, in $\{-_{N,rm,rd,wm,wd}\}$, the first argument (the length of the array) is always bound existentially outside, and the four other arguments are bound inside, unless they are bound outside; moreover, we assume that the last four arguments respect the constraints with respect to N described below.

Concrete predicates

$$\begin{aligned}
\mathit{Prod}(x, R, tw) &= \exists r : \mathsf{RId}, N : \mathbb{N}.\ 0 < N * \ulcorner x.\mathsf{N} \mapsto N \urcorner \\
&\quad * \mathsf{region}(\{\mathrm{RW}_{N,rm,rd,wm,wd}\} \cup \{\mathrm{rW}_{N,rm,rd,wm,wd}\}, T_{\mathrm{circ}}, I_{\mathrm{circ}}(x, P, r), r) * [\mathrm{P}_{tw}]^r_1 \\
\mathit{Cons}(x, R, tr) &= \exists r : \mathsf{RId}, N : \mathbb{N}.\ 0 < N * \ulcorner x.\mathsf{N} \mapsto N \urcorner \\
&\quad * \mathsf{region}(\{\mathrm{RW}_{N,rm,rd,wm,wd}\} \cup \{\mathrm{rW}_{N,rm,rd,wm,wd}\}, T_{\mathrm{circ}}, I_{\mathrm{circ}}(x, P, r), r) * [\mathrm{C}_{tr}]^r_1
\end{aligned}$$

The producer knows that the region is in a W state, because it has the exclusive right to put it in the w state, and put restores the W state. Symmetrically, the consumer knows that the region is in an R state.

We define a "distance in the circular buffer":

$$\delta_N(w, r) = \text{if } w \geq r \text{ then } w - r \text{ else } w + N - r$$

States of the protocol:

$$\begin{aligned}
&\{\mathrm{RW}_{N,rm,rd,wm,wd} | 0 \leq rm < N \wedge 0 \leq wm < N \wedge 0 \leq rd < \delta_N(wm, rm) \wedge 0 \leq wd < \delta_N(rm, wm)\} \\
\cup\ &\{\mathrm{rW}_{N,rm,rd,wm,wd} | 0 \leq rm < N \wedge 0 \leq wm < N \wedge 0 \leq rd < \delta_N(wm, rm) - 1 \wedge 0 \leq wd < \delta_N(rm, wm)\} \\
\cup\ &\{\mathrm{Rw}_{N,rm,rd,wm,wd} | 0 \leq rm < N \wedge 0 \leq wm < N \wedge 0 \leq rd < \delta_N(wm, rm) \wedge 0 \leq wd < \delta_N(rm, wm) - 1\} \\
\cup\ &\{\mathrm{rw}_{N,rm,rd,wm,wd} | 0 \leq rm < N \wedge 0 \leq wm < N \wedge 0 \leq rd < \delta_N(wm, rm) - 1 \wedge 0 < wd < \delta_N(rm, wm) - 1\}
\end{aligned}$$

The RW states are the "proper" states, where all the item in the buffer are advertised, whereas the other states are "temporary" states: in Rw, the last item in the buffer has not been advertised yet; in rW, the first item in the buffer has been removed, but but this hasn't been advertised; in rw, the last item in the buffer has not been advertised yet, and the first item has been taken away, but this hasn't been advertised.

Transitions of the protocol:

The producer can add (but not yet advertise) a new value:

$$\mathrm{P}_t : \mathrm{RW}_{N,rm,rd,wm,wd} \rightarrow \mathrm{Rw}_{N,rm,rd,wm,wd} \qquad \text{if } wd < \delta_N(rm, wm)$$
$$\mathrm{P}_t : \mathrm{rW}_{N,rm,rd,wm,wd} \rightarrow \mathrm{rw}_{N,rm,rd,wm,wd} \qquad \text{if } wd < \delta_N(rm, wm)$$

The producer can advertise a new value:

$$\mathrm{P}_t : \mathrm{Rw}_{N,rm,rd,wm,wd} \rightarrow \mathrm{RW}_{N,rm,rd,wm,wd+1}$$
$$\mathrm{P}_t : \mathrm{rw}_{N,rm,rd,wm,wd} \rightarrow \mathrm{rW}_{N,rm,rd,wm,wd+1}$$

The consumer can take (but not yet advertise availability of the slot) a value:

$$\mathrm{C}_t : \mathrm{RW}_{N,rm,rd,wm,wd} \rightarrow \mathrm{rW}_{N,rm,rd,wm,wd} \qquad \text{if } rd < \delta_N(wm, rm)$$
$$\mathrm{C}_t : \mathrm{Rw}_{N,rm,rd,wm,wd} \rightarrow \mathrm{rw}_{N,rm,rd,wm,wd} \qquad \text{if } rd < \delta_N(wm, rm)$$

The consumer can advertise availability of a slot:

$$\mathrm{C}_t : \mathrm{rW}_{N,rm,rd,wm,wd} \rightarrow \mathrm{RW}_{N,rm,rd+1,wm,wd}$$
$$\mathrm{C}_t : \mathrm{rw}_{N,rm,rd,wm,wd} \rightarrow \mathrm{Rw}_{N,rm,rd+1,wm,wd}$$

The consumer can realise that some values have been published:

$$\mathrm{C}_t : \mathrm{RW}_{N,rm,rd,wm,wd} \rightarrow \mathrm{RW}_{N,rm,rd,wm',wd'} \qquad \text{if } wm + wm = wm' + wm' \wedge wd' \leq wd$$
$$\mathrm{C}_t : \mathrm{Rw}_{N,rm,rd,wm,wd} \rightarrow \mathrm{Rw}_{N,rm,rd,wm',wd'} \qquad \text{if } wm + wm = wm' + wm' \wedge wd' \leq wd$$

The producer can realise that some values have been taken:

$$\mathrm{P}_t : \mathrm{RW}_{N,rm,rd,wm,wd} \rightarrow \mathrm{RW}_{N,rm',rd',wm,wd} \qquad \text{if } rm + rd = rm' + rd' \wedge rd < rd'$$
$$\mathrm{P}_t : \mathrm{rW}_{N,rm,rd,wm,wd} \rightarrow \mathrm{rW}_{N,rm',rd',wm,wd} \qquad \text{if } rm + rd = rm' + rd' \wedge rd < rd'$$

Interpretation of the states:

$$available(x, n, P) = \exists v.\ x.\mathsf{buf}[n] \mapsto v * P(v)$$
$$discarded(x, n) = \exists v.\ x.\mathsf{buf}[n] \mapsto v$$

$$\begin{aligned} f(x, N, P, n, m) &= \text{if } m = 0 \text{ then } \ulcorner x.\mathsf{wi} \mapsto n\%N \urcorner \\ &\quad \text{else } \ulcorner x.\mathsf{wi} \mapsto n\%N \urcorner\, \mathcal{U}_{tw}\ (\ulcorner available(x, n\%N, P) \urcorner * f(x, N, P, n+1, m-1)) \\ g(x, N, P, n, m) &= \text{if } m = 0 \text{ then } \mathsf{emp} \\ &\quad \text{else } \ulcorner x.\mathsf{wi} \mapsto n\%N \urcorner\, \mathcal{U}_{tw}\ (\ulcorner available(x, n\%N, P) \urcorner * g(x, N, P, n+1, m-1)) \end{aligned}$$

$$\begin{aligned} riupdates(x, rm, rm+rd, N) &= (\!|(x.\mathsf{ri} \mapsto (rm)\%N)\, \mathcal{U}_{tr}\ \ldots \mathcal{U}_{tr}\ (x.\mathsf{ri} \mapsto (rm+rd)\%N))|\!) \\ allflushedavailable(x, rm+rd, wm, N) &= \circledast_{0 \le k \le \delta_N(wm,(rm+rd)\%N)}\ \ulcorner available(x, (k+rm+rd)\%N, P) \urcorner \\ alldiscarded(x, rm+rd, wm+wd, N) &= \circledast_{0 \le k \le \delta_N((rm+rd)\%N,(wm+wd)\%N)}\ \ulcorner discarded(x, (k+wm+wd)\%N) \urcorner \end{aligned}$$

We use the usual $(\circledast_{t \in \mathrm{TId} \setminus \{tw\}} [\mathrm{P}_t]_1^r)$ trick to make sure which thread is the producer (same for the consumer).

$$\begin{aligned} I_{\mathrm{circ}}(x, P, r)(\mathrm{RW}_{N,rm,rd,wm,wd}) &= \exists tw, tr : \mathsf{TId}.\ tw \neq tr * (\circledast_{t \in \mathrm{TId} \setminus \{tw\}} [\mathrm{P}_t]_1^r) * (\circledast_{t \in \mathrm{TId} \setminus \{tr\}} [\mathrm{C}_t]_1^r) \\ &\quad * allflushedavailable(x, rm+rd, wm, N) \\ &\quad * (\!|f(x, N, P, wm, wd)|\!) \\ &\quad * alldiscarded(x, rm+rd, wm+wd, N) \\ &\quad * riupdates(x, rm, rm+rd, N) \\ I_{\mathrm{circ}}(x, P, r)(\mathrm{Rw}_{N,rm,rd,wm,wd}) &= \exists tw, tr : \mathsf{TId}.\ tw \neq tr * (\circledast_{t \in \mathrm{TId} \setminus \{tw\}} [\mathrm{P}_t]_1^r) * (\circledast_{t \in \mathrm{TId} \setminus \{tr\}} [\mathrm{C}_t]_1^r) \\ &\quad * allflushedavailable(x, rm+rd, wm, N) \\ &\quad * (\!|g(x, N, P, wm, wd+1)|\!)) \\ &\quad * alldiscarded(x, rm+rd, wm+wd, N) \\ &\quad * riupdates(x, rm, rm+rd, N) \\ I_{\mathrm{circ}}(x, P, r)(\mathrm{rW}_{N,rm,rd,wm,wd}) &= \exists tw, tr : \mathsf{TId}.\ tw \neq tr * (\circledast_{t \in \mathrm{TId} \setminus \{tw\}} [\mathrm{P}_t]_1^r) * (\circledast_{t \in \mathrm{TId} \setminus \{tr\}} [\mathrm{C}_t]_1^r) \\ &\quad * allflushedavailable(x, rm+rd+1, wm, N) \\ &\quad * (\!|f(x, N, P, wm, wd)|\!) \\ &\quad * alldiscarded(x, rm+rd+1, wm+wd, N) \\ &\quad * riupdates(x, rm, rm+rd, N) \\ I_{\mathrm{circ}}(x, P, r)(\mathrm{rw}_{N,rm,rd,wm,wd}) &= \exists tw, tr : \mathsf{TId}.\ tw \neq tr * (\circledast_{t \in \mathrm{TId} \setminus \{tw\}} [\mathrm{P}_t]_1^r) * (\circledast_{t \in \mathrm{TId} \setminus \{tr\}} [\mathrm{C}_t]_1^r) \\ &\quad * allflushedavailable(x, rm+rd+1, wm, N) \\ &\quad * (\!|g(x, N, P, wm, wd+1)|\!) \\ &\quad * alldiscarded(x, rm+rd+1, wm+wd, N) \\ &\quad * riupdates(x, rm, rm+rd, N) \end{aligned}$$

$\mathrm{Rw}_{rm,rd,wm,wd}$ is the same as $\mathrm{RW}_{rm,rd,wm,wd}$ but missing $x.ri \mapsto (n+m+1)\%N$ at the end

10.4 Proof outline

CircularBuffer(int N) =
$[0 < \mathsf{N} * \ulcorner \mathbf{this}.\mathsf{buf} \mapsto \mathbf{null} * \mathbf{this}.\mathsf{ri} \mapsto \mathbf{null} * \mathbf{this}.\mathsf{wi} \mapsto \mathbf{null} \urcorner * tr \neq tw]$

(**this**.buf := new int[N];
$[0 < \mathsf{N} * \overline{\textbf{this}.\mathsf{buf}[0] \mapsto \textbf{null} * \ldots * \textbf{this}.\mathsf{buf}[\mathsf{N}-1] \mapsto \textbf{null}} * \ldots ri, wi * tr \neq tw]$
this.N := N;
this.ri := 0;
this.wi := 0;
fence;
$[0 < \mathsf{N} * \ulcorner \textbf{this}.\mathsf{buf}[0] \mapsto \textbf{null} * \ldots * \textbf{this}.\mathsf{buf}[\mathsf{N}-1] \mapsto \textbf{null} * \textbf{this}.\mathsf{N} \mapsto N * \textbf{this}.\mathsf{ri} \mapsto 0 * \textbf{this}.\mathsf{wi} \mapsto 0 \urcorner * tr \neq tw]$
$[0 < \mathsf{N} * \ulcorner (\exists v.\ \textbf{this}.\mathsf{buf}[0] \mapsto v) * \ldots * (\exists v.\ \textbf{this}.\mathsf{buf}[\mathsf{N}-1] \mapsto v) * \textbf{this}.\mathsf{N} \mapsto N * \textbf{this}.\mathsf{ri} \mapsto 0 * \textbf{this}.\mathsf{wi} \mapsto 0 \urcorner * tr \neq tw]$
$[\mathit{Prod}(\textbf{this}, P, tw) * \mathit{Cons}(\textbf{this}, P, tr)]$
this)
}

Note: when the producer reads r, it *moves* the region to $\mathrm{RW}_{\mathsf{r},_,_,_}$ or $\mathrm{rW}_{\mathsf{r},_,_,_}$. Symmetrically, when the consumer reads w, it moves the region to $\mathrm{RW}_{_,_,\mathsf{w},_}$ (or $\mathrm{Rw}_{_,_,\mathsf{w},_}$).

void put(int v) =
$[\mathit{Prod}(\textbf{this}, P, tw) * \overline{P(v)} * \mathsf{iam}(tw)]$
$[0 < N * \ulcorner \textbf{this}.\mathsf{N} \mapsto N \urcorner * \mathsf{region}(\{\mathrm{RW}_{N,rm,rd,wm,wd}\} \cup \{\mathrm{rW}_{N,rm,rd,wm,wd}\}, T_{\mathrm{circ}}, I_{\mathrm{circ}}(\textbf{this}, P, r), r) * [\mathrm{P}_{tw}]^r_1 * \overline{P(v)} * \mathsf{iam}(tw)]$
let N = **this**.N **in**
$[0 < \mathsf{N} * \ulcorner \textbf{this}.\mathsf{N} \mapsto \mathsf{N} \urcorner * \mathsf{region}(\{\mathrm{RW}_{\mathsf{N},rm,rd,wm,wd}\} \cup \{\mathrm{rW}_{\mathsf{N},rm,rd,wm,wd}\}, T_{\mathrm{circ}}, I_{\mathrm{circ}}(\textbf{this}, P, r), r) * [\mathrm{P}_{tw}]^r_1 * \overline{P(v)} * \mathsf{iam}(tw)]$
let w = **this**.wi **in**
$[0 < \mathsf{N} * \ulcorner \textbf{this}.\mathsf{N} \mapsto \mathsf{N} \urcorner * \mathsf{region}(\{\mathrm{RW}_{\mathsf{N},rm,rd,wm,wd} \mid (wm+wd)\%\mathsf{N} = \mathsf{w}\} \cup \{\mathrm{rW}_{\mathsf{N},rm,rd,wm,wd} \mid (wm+wd)\%\mathsf{N} = \mathsf{w}\}, T_{\mathrm{circ}}, I_{\mathrm{circ}}(\textbf{this}, P, r), r) * [\mathrm{P}_{tw}]^r_1 * \overline{P(v)} * \mathsf{iam}(tw)]$
let r = **this**.ri **in**
$[0 < \mathsf{N} * \ulcorner \textbf{this}.\mathsf{N} \mapsto \mathsf{N} \urcorner * \mathsf{region}(\{\mathrm{RW}_{\mathsf{N},\mathsf{r},rd,wm,wd} \mid (wm+wd)\%\mathsf{N} = \mathsf{w}\} \cup \{\mathrm{rW}_{\mathsf{N},\mathsf{r},rd,wm,wd} \mid (wm+wd)\%\mathsf{N} = \mathsf{w}\}, T_{\mathrm{circ}}, I_{\mathrm{circ}}(\textbf{this}, P, r), r)) * [\mathrm{P}_{tw}]^r_1 * \overline{P(v)} * \mathsf{iam}(tw)]$
let w' = (w + 1) % N **in**
$[\ldots * \mathsf{w}' = (\mathsf{w}+1)\%N]$
if w' == r **then**
$[0 < \mathsf{N} * \ulcorner \textbf{this}.\mathsf{N} \mapsto \mathsf{N} \urcorner * \mathsf{region}(A, T_{\mathrm{circ}}, I_{\mathrm{circ}}(\textbf{this}, P, r), r) * [\mathrm{P}_{tw}]^r_1 * \overline{P(v)} * \mathsf{iam}(tw)]$
$[0 < \mathsf{N} * \ulcorner \textbf{this}.\mathsf{N} \mapsto \mathsf{N} \urcorner * \mathit{Prod}(\textbf{this}, P, tw) * \overline{P(v)} * \mathsf{iam}(tw)]$
put(i)
else
($[0 < \mathsf{N} * \ulcorner \textbf{this}.\mathsf{N} \mapsto \mathsf{N} \urcorner * \mathsf{region}(\{\mathrm{RW}_{\mathsf{N},\mathsf{r},rd,wm,wd} \mid (wm+wd)\%\mathsf{N} = \mathsf{w}\} \cup \{\mathrm{rW}_{\mathsf{N},\mathsf{r},rd,wm,wd} \mid (wm+wd)\%\mathsf{N} = \mathsf{w}\}, T_{\mathrm{circ}}, I_{\mathrm{circ}}(\textbf{this}, P, r), r) * [\mathrm{P}_{tw}]^r_1 * \overline{P(v)} * \mathsf{w}' = (\mathsf{w}+1)\%N * \mathsf{w}' \neq \mathsf{r} * \mathsf{iam}(tw)]$
this.buf[w] := v;
$[0 < \mathsf{N} * \ulcorner \textbf{this}.\mathsf{N} \mapsto \mathsf{N} \urcorner * \mathsf{region}(\{\mathrm{Rw}_{\mathsf{N},\mathsf{r},rd,wm,wd} \mid (wm+wd)\%\mathsf{N} = \mathsf{w}\} \cup \{\mathrm{rw}_{\mathsf{N},\mathsf{r},rd,wm,wd} \mid (wm+wd)\%\mathsf{N} = \mathsf{w}\}, T_{\mathrm{circ}}, I_{\mathrm{circ}}(\textbf{this}, P, r), r) * [\mathrm{P}_{tw}]^r_1 * \mathsf{w}' = (\mathsf{w}+1)\%N * \mathsf{w}' \neq \mathsf{r} * \mathsf{iam}(tw)]$
this.wi := w'
$[0 < \mathsf{N} * \ulcorner \textbf{this}.\mathsf{N} \mapsto \mathsf{N} \urcorner * \mathsf{region}(\{\mathrm{RW}_{\mathsf{N},\mathsf{r},rd,wm,wd}\} \cup \{\mathrm{rW}_{\mathsf{N},\mathsf{r},rd,wm,wd}\}, T_{\mathrm{circ}}, I_{\mathrm{circ}}(\textbf{this}, P, r), r) * [\mathrm{P}_{tw}]^r_1 * \mathsf{iam}(tw)]$
$[0 < \mathsf{N} * \ulcorner \textbf{this}.\mathsf{N} \mapsto \mathsf{N} \urcorner * \mathsf{region}(A, T_{\mathrm{circ}}, I_{\mathrm{circ}}(\textbf{this}, P, r), r) * [\mathrm{P}_{tw}]^r_1]$)
$[\mathit{Prod}(\textbf{this}, P, tw)]$

int get() =
$[\mathit{Cons}(\textbf{this}, P, tr) * \mathsf{iam}(tr)]$
$[0 < N * \ulcorner \textbf{this}.\mathsf{N} \mapsto N \urcorner * \mathsf{region}(\{\mathrm{RW}_{N,rm,rd,wm,wd}\} \cup \{\mathrm{Rw}_{N,rm,rd,wm,wd}\}, T_{\mathrm{circ}}, I_{\mathrm{circ}}(\textbf{this}, P, r), r) * [\mathrm{C}_{tr}]^r_1 * \mathsf{iam}(tr)]$
let N = **this**.N **in**
$[0 < \mathsf{N} * \ulcorner \textbf{this}.\mathsf{N} \mapsto \mathsf{N} \urcorner * \mathsf{region}(\{\mathrm{RW}_{\mathsf{N},rm,rd,wm,wd}\} \cup \{\mathrm{Rw}_{\mathsf{N},rm,rd,wm,wd}\}, T_{\mathrm{circ}}, I_{\mathrm{circ}}(\textbf{this}, P, r), r) * [\mathrm{C}_{tr}]^r_1 * \mathsf{iam}(tr)]$
let r = **this**.ri **in**
$[0 < \mathsf{N} * \ulcorner \textbf{this}.\mathsf{N} \mapsto \mathsf{N} \urcorner * \mathsf{region}(\{\mathrm{RW}_{\mathsf{N},rm,rd,wm,wd} \mid (rm+rd)\%\mathsf{N} = \mathsf{r}\} \cup \{\mathrm{Rw}_{\mathsf{N},rm,rd,wm,wd} \mid (rm+rd)\%\mathsf{N} = \mathsf{r}\}, T_{\mathrm{circ}}, I_{\mathrm{circ}}(\textbf{this}, P, r), r) * [\mathrm{C}_{tr}]^r_1 * \mathsf{iam}(tr)]$
let w = **this**.wi **in**
$[0 < \mathsf{N} * \ulcorner \textbf{this}.\mathsf{N} \mapsto \mathsf{N} \urcorner * \mathsf{region}(\{\mathrm{RW}_{\mathsf{N},rm,rd,\mathsf{w},wd} \mid (rm+rd)\%\mathsf{N} = \mathsf{r}\} \cup \{\mathrm{Rw}_{\mathsf{N},rm,rd,\mathsf{w},wd} \mid (rm+rd)\%\mathsf{N} = \mathsf{r}\}, T_{\mathrm{circ}}, I_{\mathrm{circ}}(\textbf{this}, P, r), r) * [\mathrm{C}_{tr}]^r_1 * \mathsf{iam}(tr)]$
let r' = (r + 1) % N **in**
$[\ldots * \mathsf{r}' = (\mathsf{r}+1)\%\mathsf{N}]$
if w == r' **then**
$[0 < \mathsf{N} * \ulcorner \textbf{this}.\mathsf{N} \mapsto \mathsf{N} \urcorner * \mathsf{region}(\{\mathrm{RW}_{\mathsf{N},rm,rd,wm,wd}\} \cup \{\mathrm{Rw}_{\mathsf{N},rm,rd,wm,wd}\}, T_{\mathrm{circ}}, I_{\mathrm{circ}}(\textbf{this}, P, r), r) * [\mathrm{C}_{tr}]^r_1 * \mathsf{iam}(tr)]$

$[Cons(\textbf{this}, P, tr) * \mathsf{iam}(tr)]$

get()

else

($[0 < \mathsf{N} * \ulcorner\textbf{this}.\mathsf{N} \mapsto \mathsf{N}\urcorner * \mathsf{region}(\{\mathrm{RW}_{\mathsf{N},rm,rd,\mathsf{w},wd} \mid (rm+rd)\%\mathsf{N} = \mathsf{r}\} \cup \{\mathrm{Rw}_{\mathsf{N},rm,rd,\mathsf{w},wd} \mid (rm+rd)\%\mathsf{N} = \mathsf{r}\}, T_{\mathrm{circ}}, I_{\mathrm{circ}}(\textbf{this}, P, r), r) * [\mathrm{C}_{tr}]^r_1 * \mathsf{r}' = (\mathsf{r}+1)\%\mathsf{N} * \mathsf{w} \neq \mathsf{r}' * \mathsf{iam}(tr)]$

let v = **this**.buf[r] **in**

$[0 < \mathsf{N} * \ulcorner\textbf{this}.\mathsf{N} \mapsto \mathsf{N}\urcorner * \mathsf{region}(\{\mathrm{rW}_{\mathsf{N},rm,rd,\mathsf{w},wd} \mid (rm+rd)\%\mathsf{N} = \mathsf{r}\} \cup \{\mathrm{rw}_{\mathsf{N},rm,rd,\mathsf{w},wd} \mid (rm+rd)\%\mathsf{N} = \mathsf{r}\}, T_{\mathrm{circ}}, I_{\mathrm{circ}}(\textbf{this}, P, r), r) * [\mathrm{C}_{tr}]^r_1 * \mathsf{r}' = (\mathsf{r}+1)\%\mathsf{N} * \mathsf{w} \neq \mathsf{r}' * \ulcorner P(v)\urcorner * \mathsf{iam}(tr)]$

this.ri := r';

$[0 < \mathsf{N} * \ulcorner\textbf{this}.\mathsf{N} \mapsto \mathsf{N}\urcorner * \mathsf{region}(\{\mathrm{RW}_{\mathsf{N},rm,rd,wm,wd}\} \cup \{\mathrm{Rw}_{\mathsf{N},rm,rd,wm,wd}\}, T_{\mathrm{circ}}, I_{\mathrm{circ}}(\textbf{this}, P, r), r) * [\mathrm{C}_{tr}]^r_1 * \ulcorner P(v)\urcorner * \mathsf{iam}(tr)]$

$[Cons(\textbf{this}, P, tr) * \ulcorner P(v)\urcorner]$

v)

References

[1] J. Corbet. Ticket spinlocks, Feb. 2008. Available at http://lwn.net/Articles/267968/.

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

[3] D. Howells and P. E. McKenney. Circular buffers. Available at https://www.kernel.org/doc/Documentation/circular-buffers.txt.

[4] J. M. Mellor-Crummey and M. L. Scott. Algorithms for scalable synchronization on shared-memory multiprocessors. *Transactions of Computer Systems*, 9(1):21–65, Feb. 1991.

[5] D. C. Schmidt and T. Harrison. Double-checked locking - an optimization pattern for efficiently initializing and accessing thread-safe objects, 1997. Available at http://www.dre.vanderbilt.edu/~schmidt/PDF/DC-Locking.pdf.

[6] K. Svendsen and L. Birkedal. Impredicative Concurrent Abstract Predicates. Technical report, 2013. Available at www.kasv.dk/articles/icap-tr.pdf.

[7] K. Svendsen and L. Birkedal. Impredicative Concurrent Abstract Predicates. In *Proceedings of ESOP*, 2014.

[8] K. Svendsen, L. Birkedal, and M. Parkinson. Modular Reasoning about Separation of Concurrent Data Structures. In *Proceedings of ESOP*, 2013.

[9] A. Turon, V. Vafeiadis, and D. Dreyer. GPS: Navigating Weak Memory with Ghosts, Protocols, and Separation, March 2014. Under submission.