A separation logic for a promising semantics (Technical appendix)

December 7, 2017

This is the technical appendix accompanying the article "A separation logic for a promising semantics". It contains a soundness proof for the SLR logic. The programming language and logic presented in this appendix extends the programming language and logic presented in the paper with plain accesses. The programming language has further been extended with compare-and-swap operations and the logic has been proven sound under the additional memory reduction rules required to support compare-and-swap.

Contents

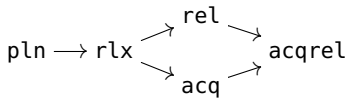
1 Operational semantics

Programming Language Let *Loc* be the set of memory locations, *Val* be the set of values, *Tid* be the set of thread identifiers, and *Reg* be the set of register identifiers.

We assume *Loc* is the union of two disjoint sets of plain and non-plain memory locations, *PLoc* and *NPLoc*, respectively. Throughout the rest of the document we implicitly assume that all plain accesses are performed on plain locations and all non-plain accesses on non-plain locations.

$$
\begin{array}{rcl}
e \in \mathit{Expr} & ::= & r \mid e_1 + e_2 \mid \ldots \\
s \in \mathit{Stm} & ::= & r = e \\
& \mid & [e]_{\mathtt{pln}} := r \mid [e]_{\mathtt{rlx}} := r \mid [e]_{\mathtt{rel}} := r \\
& \mid & r := [e]_{\mathtt{pln}} \mid r := [e]_{\mathtt{rlx}} \mid r := [e]_{\mathtt{acq}} \\
& \mid & r_1, r_2 = \mathtt{cas}_{\mathtt{rlx}}(e, r_3, r_4) \\
& \mid & r_1, r_2 = \mathtt{cas}_{\mathtt{rel}}(e, r_3, r_4) \\
& \mid & r_1, r_2 = \mathtt{cas}_{\mathtt{acq}}(e, r_3, r_4) \\
& \mid & r_1, r_2 = \mathtt{cas}_{\mathtt{acqrel}}(e, r_3, r_4) \\
& \mid & \textbf{if } e \textbf{ then } s_1 \textbf{ else } s_2 \\
& \mid & \textbf{while } e \textbf{ do } s \\
& \mid & s_1; s_2 \mid \textbf{skip}
\end{array}
$$

The modes for accesses are partially ordered by $\sqsubset$ as follows:



Thread store $\boxed{\mu \in \mathit{TStore} = \mathit{Reg} \to \mathit{Val}}$

A *thread store*, μ, is a function assigning a value to every register identifiers.

Thread local state $\boxed{\sigma \in \mathit{TLState} = \mathit{TStore} \times \mathit{Stm}}$

A *thread local state*, $\sigma = (\mu, s)$, consists of a thread store and a statement (to be executed by the thread).

Memory actions $\boxed{a \in \mathit{MO}}$

Transitions are labelled with *memory actions*:

$$a \in \mathit{MO} \quad ::= \quad \tau \mid \mathsf{W}_o\, x\, v \mid \mathsf{R}_o\, x\, v \mid \mathsf{U}_o\, x\, v_1\, v_2$$

τ, which stands for a silent action (as opposed to a proper memory action), is elided.

Thread-local state reduction $\boxed{\sigma \xrightarrow{a} \sigma'}$

$$\frac{[\![e]\!](\mu) = \top}{(\mu, \textsf{if } e \textsf{ then } s_1 \textsf{ else } s_2) \longrightarrow (\mu, s_1)} \qquad \frac{[\![e]\!](\mu) = \bot}{(\mu, \textsf{if } e \textsf{ then } s_1 \textsf{ else } s_2) \longrightarrow (\mu, s_2)} \qquad \frac{[\![e]\!](\mu) = \bot}{(\mu, \textsf{while } e \textsf{ do } s) \longrightarrow (\mu, \textsf{skip})}$$

$$\frac{[\![e]\!](\mu) = \top}{(\mu, \textsf{while } e \textsf{ do } s) \longrightarrow (\mu, s; \textsf{while } e \textsf{ do } s)} \qquad \frac{}{(\mu, \textsf{skip}; s) \longrightarrow (\mu, s)} \qquad \frac{(\mu, s_1) \xrightarrow{a} (\mu', s_1')}{(\mu, s_1; s_2) \xrightarrow{a} (\mu', s_1'; s_2)}$$

$$\frac{[\![e]\!](\mu) = v}{(\mu, r = e) \longrightarrow (\mu[r \mapsto v], \textsf{skip})} \qquad (\mu, [x]_o := e) \xrightarrow{\mathsf{W}_o\, x\, \mu(e)} (\mu, \textsf{skip}) \qquad (\mu, r := [x]_o) \xrightarrow{\mathsf{R}_o\, x\, v} (\mu[r \mapsto v], \textsf{skip})$$

$$\frac{o = \texttt{rlx} \vee o = \texttt{rel} \Rightarrow o' = \texttt{rlx} \qquad o = \texttt{acq} \vee o = \texttt{acqrel} \Rightarrow o' = \texttt{acq}}{(\mu, r_3, r_4 = \texttt{cas}_o(x, e_1, e_2)) \xrightarrow{\mathsf{R}_{o'}\, x\, v} (\mu[r_3 \mapsto 0, r_4 \mapsto v], \textsf{skip})}$$

$$(\mu, r_3, r_4 = \texttt{cas}_o(x, e_1, e_2)) \xrightarrow{\mathsf{U}_o\, x\, \mu(e_1)\, \mu(e_2)} (\mu[r_3 \mapsto 1, r_4 \mapsto v_1], \textsf{skip})$$

Timestamp $\boxed{t \in \mathit{Time}}$

Time is an infinite set of *timestamps*, densely totally ordered by $\leq$, with 0 being the minimum element.

Timemap $\boxed{T \in \mathit{Timemap} = \mathit{Loc} \to \mathit{Time}}$

A *timemap* is a function $T : \mathit{Loc} \to \mathit{Time}$. The order $\leq$ is extended pointwise to timemaps.

View $\boxed{V \in \mathit{View} \subseteq \mathit{Timemap} \times \mathit{Timemap}}$

A *view* is a pair $V = \langle T_{\texttt{pln}}, T_{\texttt{rlx}} \rangle$ of timemaps satisfying $T_{\texttt{pln}} \leq T_{\texttt{rlx}}$. We denote by $V.\texttt{pln}$ and $V.\texttt{rlx}$ the components of V.

Additional notations for timemaps and views $\bot$ and $\sqcup$ denote the natural bottom elements and join operations for timemaps and for views (pointwise extensions of the initial timestamp 0 and the $\sqcup$—*i.e.,* max—operation on timestamps); $\{x@t\}$ denotes the timemap assigning t to x and 0 to other locations. We write $T_\bot$ for the least timemap that maps every location to 0 and $V_\bot$ for the view $\langle T_\bot, T_\bot \rangle$.

Message $\boxed{m \in \mathit{Msg} \subseteq \mathit{Loc} \times \mathit{Val} \times \mathit{Time} \times \mathit{Time} \times \mathit{View} \times \mathit{Tid} \times \mathit{Mod}}$

A *message* is a tuple $m = \langle x :^o_i v, R@(f,t] \rangle$, where $x \in \mathit{Loc}$, $v \in \mathit{Val}$, $f, t \in \mathit{Time}$, $R \in \mathit{View}$, $i \in \mathit{Tid}$, and $o \in \mathit{Mod}$, such that (*i*) $f < t$ or $f = t = 0$; (*ii*) $R.\texttt{rlx}(x) = R.\texttt{pln}(x) = t$; and (*iii*) $o = \texttt{pln}$ iff $x \in \mathit{PLoc}$. We denote by $m.\texttt{loc}$, $m.\texttt{val}$, $m.\texttt{from}$, $m.\texttt{time}$, $m.\texttt{view}$, and $m.\texttt{mod}$ the components of m. Two messages $m = \langle x :^o_i v, R@(f,t] \rangle$ and $m' = \langle x' :^{o'}_j v', R'@(f',t'] \rangle$ are called *disjoint*, denoted $m \# m'$, if either $x \neq x'$, $t \leq f' < t$, or $t' \leq f < t$.

Memory $\boxed{M \in \mathit{Mem} \subseteq \mathcal{P}_{\mathit{fin}}(\mathit{Msg})}$

A *memory* is a (nonempty) pairwise disjoint finite set of messages. A memory M supports the following insertions of a message $m = \langle x :^o_i v, R@(f,t] \rangle$:

- The *additive insertion*, denoted by $M \overset{\text{A}}{\hookleftarrow} m$, is only defined if $\{m\} \# M$, in which case it is given by $\{m\} \cup M$.
- The *splitting insertion*, denoted by $M \overset{\text{S}}{\hookleftarrow} m$, is only defined if there exists $m' = \langle x :^o_i v', R'@(f,t'] \rangle$ with $t < t'$ in M, in which case it is given by $M \backslash \{m'\} \cup \{m, \langle x :^o_i v', R'@(t,t'] \rangle\}$.

- The *lowering insertion*, denoted by $M \overset{\text{U}}{\hookleftarrow} m$, is only defined if there exists $m' = \langle x :^{o}_{i} v, R'@(f,t]\rangle$ with $R \leq R'$ in M, in which case it is given by $M \backslash \{m'\} \cup \{m\}$.

Notation for restricting a memory:

$$
\begin{aligned}
M(i) &\stackrel{def}{=} \{m \in M \mid m.\texttt{tid} = i\} \\
M(x) &\stackrel{def}{=} \{m \in M \mid m.\texttt{loc} = x\} \\
M(i,x) &\stackrel{def}{=} M(i) \cap M(x) \\
M(\texttt{rel}) &\stackrel{def}{=} \{m \in M \mid m.\texttt{mod} = \texttt{rel}\} \\
M(\texttt{rlx}) &\stackrel{def}{=} \{m \in M \mid m.\texttt{mod} = \texttt{rlx}\} \\
M(\texttt{rel},x) &\stackrel{def}{=} M(\texttt{rel}) \cap M(x) \\
M(\texttt{rlx},x) &\stackrel{def}{=} M(\texttt{rlx}) \cap M(x)
\end{aligned}
$$

Memory reduction $\boxed{\langle M, P\rangle \xrightarrow{m} \langle M', P'\rangle}$

$$\text{MEMORY: NEW}\quad \frac{}{\langle M,P\rangle \xrightarrow{m} \langle M \overset{\text{A}}{\hookleftarrow} m, P\rangle} \qquad \text{MEMORY: FULFILL}\quad \frac{\hookleftarrow \in \{\overset{\text{S}}{\hookleftarrow}, \overset{\text{U}}{\hookleftarrow}\} \qquad P' = P \hookleftarrow m \qquad M' = M \hookleftarrow m}{\langle M,P\rangle \xrightarrow{m} \langle M', P' \setminus \{m\}\rangle}$$

Closed memory Given a timemap T and a memory M, we write $T \in M$ if, for every $x \in \mathit{Loc}$, we have $T(x) = m.\texttt{time}$ for some $m \in M$ with $m.\texttt{loc} = x$. For a view V, we write $V \in M$ if $T \in M$ for each component timemap T of V. A memory M is *closed* if $m.\texttt{view} \in M$ for every $m \in M$.

Future memory For memories M, M', we write $M \to M'$ if $M' \in \{M \overset{\text{A}}{\hookleftarrow} m, M \overset{\text{S}}{\hookleftarrow} m, M \overset{\text{U}}{\hookleftarrow} m\}$ for some message m, and M' is closed. We say that M' is a *future memory* of M w.r.t. a memory P, if $P \subseteq M'$ and $M \to^* M'$.

Global Configuration $\boxed{\mathit{gconf} \in \mathit{GConf} = \mathit{Mem} \times \mathit{Mem}}$

A *global configuration* is a tuple $\mathit{gconf} = \langle M, P\rangle$,where M is a memory and $P \subseteq M$ is a memory called *promise memory*. We denote by $\mathit{gconf}.\texttt{M}$ and $\mathit{gconf}.\texttt{P}$ the components of gconf. We write $\mathit{gconf}_{\perp}$ to denote the empty global configuration, $\langle \emptyset, \emptyset\rangle$.

Thread state $\boxed{\mathit{TS} \in \mathit{TLState} \times \mathit{View}}$

A *thread state* is a pair $\mathit{TS} = \langle \sigma, V\rangle$, where σ is a thread local state and V is a view. We denote by $\mathit{TS}.\sigma$ and $\mathit{TS}.V$ the components of TS.

Thread configuration $\boxed{\langle \mathit{TS}, \langle M, P\rangle\rangle}$

A *thread configuration* is a tuple $\langle \mathit{TS}, \langle M, P\rangle\rangle$, where TS is a thread state, and $\langle M, P\rangle$ is a global configuration.

Transition mode $\boxed{\beta \in \mathit{TM}}$

$$\beta \in \mathit{TM} \quad ::= \quad \text{NP} \mid \text{promise}$$

Given a β-labelled relation $\to$, $a \to b$ stands for $\exists \beta.\ a \xrightarrow{\beta} b$.

Thread configuration reduction $\boxed{\langle TS, \langle M, P\rangle\rangle \xrightarrow{\beta}_i \langle TS', \langle M', P'\rangle\rangle}$

THREAD: SILENT

$$\frac{\sigma \longrightarrow \sigma'}{\langle\langle \sigma, V\rangle, \langle M, P\rangle\rangle \xrightarrow{\text{NP}}_i \langle\langle \sigma', V\rangle, \langle M, P\rangle\rangle}$$

THREAD: READ

$$\frac{\begin{array}{c}\sigma \xrightarrow{\mathsf{R}_o\, x\, v} \sigma' \qquad m = \langle x :^{o_\mathrm{r}}_j v, R@(\_, t]\rangle \in M \\ o = \mathtt{pln} \Longrightarrow V.\mathtt{pln}(x) \le t \ \wedge\ V' = V \sqcup [\mathtt{pln} : \bot, \mathtt{rlx} : \{x@t\}] \\ o = \mathtt{rlx} \Longrightarrow V.\mathtt{rlx}(x) \le t \ \wedge\ V' = V \sqcup [\mathtt{pln} : \{x@t\}, \mathtt{rlx} : \{x@t\}] \\ o = \mathtt{acq} \Longrightarrow V.\mathtt{rlx}(x) \le t \ \wedge\ V' = V \sqcup [\mathtt{pln} : \{x@t\}, \mathtt{rlx} : \{x@t\}] \sqcup R\end{array}}{\langle\langle \sigma, V\rangle, \langle M, P\rangle\rangle \xrightarrow{\text{NP}}_i \langle\langle \sigma', V'\rangle, \langle M, P\rangle\rangle}$$

THREAD: WRITE

$$\frac{\begin{array}{c}\sigma \xrightarrow{\mathsf{W}_o\, x\, v} \sigma' \qquad \langle M, P\rangle \xrightarrow{\langle x :^o_i v, R@(\_, t]\rangle} \langle M', P'\rangle \qquad V.\mathtt{rlx}(x) < t \\ V' = V \sqcup [\mathtt{pln} : \{x@t\}, \mathtt{rlx} : \{x@t\}] \\ o \sqsubseteq \mathtt{rlx} \Longrightarrow R = [\mathtt{pln} : \{x@t\}, \mathtt{rlx} : \{x@t\}] \\ o = \mathtt{rel} \Longrightarrow R = V' \ \wedge\ P(i, x) = \emptyset\end{array}}{\langle\langle \sigma, V\rangle, \langle M, P\rangle\rangle \xrightarrow{\text{NP}}_i \langle\langle \sigma', V'\rangle, \langle M', P'\rangle\rangle}$$

THREAD: UPDATE

$$\frac{\begin{array}{c}\sigma \xrightarrow{\mathsf{U}_o\, x\, v_\mathrm{r}\, v_\mathrm{w}} \sigma' \qquad m = \langle x :^{o_\mathrm{r}}_j v_\mathrm{r}, R_\mathrm{r}@(\_, f]\rangle \in M \\ \langle M, P\rangle \xrightarrow{\langle x :^o_i v_\mathrm{w}, R_\mathrm{w}@(f, t]\rangle} \langle M', P'\rangle \qquad V.\mathtt{rlx}(x) \le f \\ o \sqsubseteq \mathtt{rel} \Longrightarrow V' = V \sqcup [\mathtt{pln} : \{x@t\}, \mathtt{rlx} : \{x@t\}] \\ o \sqsupseteq \mathtt{acq} \Longrightarrow V' = V \sqcup [\mathtt{pln} : \{x@t\}, \mathtt{rlx} : \{x@t\}] \sqcup R_\mathrm{r} \\ o \sqsubseteq \mathtt{acq} \Longrightarrow R_\mathrm{w} = R_\mathrm{r} \sqcup [\mathtt{pln} : \{x@t\}, \mathtt{rlx} : \{x@t\}] \\ o \sqsupseteq \mathtt{rel} \Longrightarrow R_\mathrm{w} = V \sqcup [\mathtt{pln} : \{x@t\}, \mathtt{rlx} : \{x@t\}] \sqcup R_\mathrm{r} \\ o \sqsupseteq \mathtt{rel} \Longrightarrow P(i, x) = \emptyset\end{array}}{\langle\langle \sigma, V\rangle, \langle M, P\rangle\rangle \xrightarrow{\text{NP}}_i \langle\langle \sigma', V'\rangle, \langle M', P'\rangle\rangle}$$

THREAD: PROMISE

$$\frac{m = \langle x :^o_i \_, R@(\_, t]\rangle \qquad \hookleftarrow \in \{\overset{\text{A}}{\hookleftarrow}, \overset{\text{S}}{\hookleftarrow}, \overset{\text{U}}{\hookleftarrow}\} \qquad P' = P \hookleftarrow m \qquad M' = M \hookleftarrow m \qquad R = [\mathtt{pln} : \{x@t\}, \mathtt{rlx} : \{x@t\}]}{\langle TS, \langle M, P\rangle\rangle \xrightarrow{\text{promise}}_i \langle TS, \langle M', P'\rangle\rangle}$$

Consistency A thread configuration $\langle TS, \langle M, P\rangle\rangle$ is called *consistent* w.r.t. $i \in Tid$ if for every future memory M_future of M w.r.t. $P(i)$ we have $\langle TS, \langle M_\text{future}, P\rangle\rangle \xrightarrow{\text{NP}^*}_i \langle TS', \langle M', P'\rangle\rangle$ for some TS', M', P' such that $P'(i) = \emptyset$.

Thread certified reduction $\boxed{\langle TS, \langle M, P\rangle\rangle \xRightarrow{\beta}_i \langle TS', \langle M', P'\rangle\rangle}$

$$\frac{\langle TS, \langle M, P\rangle\rangle \xrightarrow{\text{promise}}_i \langle TS, \langle M', P'\rangle\rangle \qquad \langle TS, \langle M', P'\rangle\rangle \text{ is consistent w.r.t. } i}{\langle TS, \langle M, P\rangle\rangle \xRightarrow{\text{promise}}_i \langle TS, \langle M', P'\rangle\rangle}$$

$$\frac{\langle TS, \langle M, P\rangle\rangle \xrightarrow{\text{NP}^+}_i \langle TS', \langle M', P'\rangle\rangle \qquad \langle TS', \langle M', P'\rangle\rangle \text{ is consistent w.r.t. } i}{\langle TS, \langle M, P\rangle\rangle \xRightarrow{\text{NP}}_i \langle TS', \langle M', P'\rangle\rangle}$$

Machine state $\boxed{\mathbf{MS} = \langle \mathcal{TS}, \langle M, P\rangle\rangle}$

A *machine state* is a tuple $\mathbf{MS} = \langle \mathcal{TS}, \langle M, P\rangle\rangle$, where $\mathcal{TS}$ is a function assigning a thread state to every thread, and $\langle M, P\rangle$ is a global configuration. The initial state $\mathbf{MS}^0$ (for a given program) consists of the function $\mathcal{TS}^0$ mapping each thread i to its initial state $\langle \sigma_i^0, \bot\rangle$ (where σ_i^0 is the thread's initial local state, and $\bot$ is the zero view (all timestamps in timemaps are 0), the empty set of promises; the initial memory M^0 consisting of one message $\langle x :^{\mathtt{rlx}}_0 0, \bot@(0, 0]\rangle$ for each location x.

Machine reduction $\boxed{\langle \mathcal{TS}, \langle M, P\rangle\rangle \overset{\beta}{\Longrightarrow}_i \langle \mathcal{TS}', \langle M', P'\rangle\rangle}$

$$\frac{\langle \mathcal{TS}(i), \langle M, P\rangle\rangle \overset{\text{promise}}{\Longrightarrow}_i \langle \mathcal{TS}(i), \langle M', P'\rangle\rangle}{\langle \mathcal{TS}, \langle M, P\rangle\rangle \overset{\text{promise}}{\Longrightarrow}_i \langle \mathcal{TS}, \langle M', P'\rangle\rangle} \qquad \frac{\langle \mathcal{TS}(i), \langle M, P\rangle\rangle \overset{\text{NP}}{\Longrightarrow}_i \langle TS', \langle M', P'\rangle\rangle}{\langle \mathcal{TS}, \langle M, P\rangle\rangle \overset{\text{NP}}{\Longrightarrow}_i \langle \mathcal{TS}[i \mapsto TS'], \langle M', P'\rangle\rangle}$$

Promise certifications satisfy the following basic property: making a promise and then certifying it leads to a state that is reachable without performing any promises.

Lemma 1. Whenever $\langle\langle \sigma, V\rangle, \langle M, P\rangle\rangle \overset{\text{promise}}{\longrightarrow}_i \langle\langle \sigma, V\rangle, \langle M', P'\rangle\rangle \overset{\text{NP}}{\longrightarrow}_i{}^{*} \langle\langle \sigma', V'\rangle, \langle M'', P''\rangle\rangle$ and $P''(i) = \emptyset$, then $\langle\langle \sigma, V\rangle, \langle M, P\rangle\rangle \overset{\text{NP}}{\longrightarrow}_i{}^{*} \langle\langle \sigma', V'\rangle, \langle M'', P''\rangle\rangle$.

1.1 Simplified operational semantics

Memory reduction $\boxed{\langle M, P\rangle \overset{m}{\longrightarrow} \langle M', P'\rangle}$

$$\frac{}{\langle M, P\rangle \overset{m}{\longrightarrow} \langle M \overset{\text{A}}{\hookleftarrow} m, P\rangle} \qquad \frac{m \in P}{\langle M, P\rangle \overset{m}{\longrightarrow} \langle M, P \setminus \{m\}\rangle}$$

Thread: Silent
$$\frac{\sigma \longrightarrow \sigma'}{\langle\langle \sigma, V\rangle, \langle M, P\rangle\rangle \overset{\text{NP}}{\longrightarrow}_i \langle\langle \sigma', V\rangle, \langle M, P\rangle\rangle}$$

Thread: Read
$$\frac{\begin{array}{c}\sigma \overset{\mathsf{R}_o\, x\, v}{\longrightarrow} \sigma' \qquad \langle x :_j^{o_r} v, R@t]\rangle \in M \\ V(x) \le t \\ o = \mathtt{rlx} \Longrightarrow V' = V[x \mapsto t] \\ o = \mathtt{acq} \Longrightarrow V' = V[x \mapsto t] \sqcup R\end{array}}{\langle\langle \sigma, V\rangle, \langle M, P\rangle\rangle \overset{\text{NP}}{\longrightarrow}_i \langle\langle \sigma', V'\rangle, \langle M, P\rangle\rangle}$$

Thread: Write
$$\frac{\begin{array}{c}\sigma \overset{\mathsf{W}_o\, x\, v}{\longrightarrow} \sigma' \qquad \langle M, P\rangle \overset{\langle x :_i^o v, R@t]\rangle}{\longrightarrow} \langle M', P'\rangle \\ V(x) < t \qquad V' = V[x \mapsto t] \\ o = \mathtt{rlx} \Longrightarrow R = \{x@t\} \\ o = \mathtt{rel} \Longrightarrow R = V' \ \wedge\ P(i, x) = \emptyset\end{array}}{\langle\langle \sigma, V\rangle, \langle M, P\rangle\rangle \overset{\text{NP}}{\longrightarrow}_i \langle\langle \sigma', V'\rangle, \langle M', P'\rangle\rangle}$$

Thread: Promise
$$\frac{\begin{array}{c}m = \langle x :_i^{\mathtt{rlx}} v, R@t]\rangle \\ M' = M \overset{\text{A}}{\hookleftarrow} m \qquad P' = P \overset{\text{A}}{\hookleftarrow} m\end{array}}{\langle TS, \langle M, P\rangle\rangle \overset{\text{prom}}{\longrightarrow}_i \langle TS, \langle M', P'\rangle\rangle}$$

2 Program logic

The SLR assertion language is generated by the following grammar.

$$\begin{array}{rcl} P, Q \in \mathit{Assn} & ::= & \bot \mid \top \mid P \vee Q \mid P \wedge Q \mid P \Rightarrow Q \mid \forall x.\ P \mid \exists x.\ P \mid N_1 = N_2 \mid \phi(N) \\ & \mid & \top \mid P * Q \mid \mathsf{Rel}(\ell, \phi) \mid \mathsf{Acq}(\ell, \phi) \mid \mathsf{O}(\ell, v, t) \mid \mathsf{W}^{\pi}(\ell, X) \mid \ell \overset{\pi}{\mapsto} v \mid \nabla P \mid \mathrm{pure}(P) \\ \phi \in \mathit{Pred} & ::= & \lambda x.\ P \end{array}$$

Here M, ℓ, v, t, π and X all range over a simply-typed term language which we assume includes booleans, locations, values and expressions of the programming language, fractional permissions, and timestamps, and is closed under paring, finite sets and sequences. By convention we assume that ℓ, v, t, π and X range over terms of type location, value, timestamp, permission and sets of pairs of values and timestamps, respectively.

2.1 Assertion logic

The judgments of the assertion logic include an entailment judgment, $P \vdash Q$, a view shift judgment, $\vdash P \Rrightarrow Q$. The entailment judgment includes all the usual rules of first-order separation logic, which we elide. In addition it includes a number of axioms about SLR assertions, which are given below.

$$\mathsf{Acq}(x, \lambda v.\ \phi_1(v) * \phi_2(v)) \Leftrightarrow \mathsf{Acq}(x, \phi_1) * \mathsf{Acq}(x, \phi_2)$$
$$\mathsf{Rel}(x, \phi) \Leftrightarrow \mathsf{Rel}(x, \phi) * \mathsf{Rel}(x, \phi)$$

$$\mathsf{W}^{\pi}(x, X) * (v, t) \in X \Rrightarrow \mathsf{W}^{\pi}(x, X) * \mathsf{O}(x, v, t)$$
$$\mathsf{W}^{1}(x, X) * \mathsf{O}(x, a, t) \Rrightarrow \mathsf{W}^{1}(x, X) * \mathsf{O}(x, a, t) * (a, t) \in X$$
$$\mathsf{W}^{\pi}(x, X) * (v, t) \in X * (v', t') \in X * v \neq v' \Rrightarrow \mathsf{W}^{\pi}(x, X) * t \neq t'$$
$$\mathsf{W}^{\pi}(x, X) * (\_, t) \in X * (\_, t') \in X \Rrightarrow \mathsf{W}^{\pi}(x, X) * t < t' \vee t = t' \vee t' < t$$

$$\mathsf{W}_{\pi_1 + \pi_2}(x, X_1 \cup X_2) \Leftrightarrow \mathsf{W}_{\pi_1}(x, X_1) * \mathsf{W}_{\pi_2}(x, X_2)$$

$$x \overset{\pi_1 + \pi_2}{\mapsto} v \Leftrightarrow x \overset{\pi_1}{\mapsto} v * x \overset{\pi_2}{\mapsto} v$$
$$x \overset{\pi_1}{\mapsto} v_1 * x \overset{\pi_2}{\mapsto} v_2 \Rightarrow x \overset{\pi_1}{\mapsto} v_1 * x \overset{\pi_2}{\mapsto} v_2 * v_1 = v_2$$

For all assertions P expressible in the first-order logic fragment of our assertion logic, we have $\vdash \mathrm{pure}(P)$.

2.2 Specification logic

The judgment of the specification logic is $\vdash \{P\}\ s\ \{Q\}$.

$$\frac{\vdash P \Rrightarrow P' \qquad \vdash \{P'\}\, s\, \{Q'\} \qquad \vdash Q' \Rrightarrow Q}{\vdash \{P\}\, s\, \{Q\}} \qquad \frac{\vdash \{P\}\, s\, \{Q\} \qquad \mathit{mod}(s) \cap \mathit{FRV}(R) = \emptyset}{\vdash \{P * R\}\, s\, \{Q * R\}}$$

$$\frac{\{P\}\ s_1\ \{Q\} \qquad \{Q\}\ s_2\ \{R\}}{\{P\}\ s_1; s_2\ \{R\}} \qquad \frac{\{P \wedge e = \top\}\ s\ \{P\}}{\{P\}\ \textbf{while}\ e\ \textbf{do}\ s\ \{P \wedge e = \bot\}}$$

$$\frac{\{P \wedge e = \top\}\ s_1\ \{Q\} \qquad \{P \wedge e = \bot\}\ s_2\ \{Q\}}{\{P\}\ \textbf{if}\ e\ \textbf{then}\ s_1\ \textbf{else}\ s_2\ \{Q\}} \qquad \frac{\{P\}\ s\ \{Q\}}{\{\exists x.\ P\}\ s\ \{\exists x.\ Q\}}$$

$$\frac{\vdash P \Rightarrow Q}{\vdash P \Rrightarrow Q} \qquad \frac{\vdash P \Rrightarrow Q}{\vdash P * R \Rrightarrow Q * R} \qquad \frac{\vdash P \Rrightarrow Q \qquad \vdash Q \Rrightarrow R}{\vdash P \Rrightarrow R}$$

Plain accesses

$$\vdash \left\{x = e * x \overset{1}{\mapsto} \_\right\} [e]_{\mathtt{pln}} := a \left\{x \overset{1}{\mapsto} a\right\}$$
$$\vdash \left\{x = e * x \overset{\pi}{\mapsto} v\right\} a := [e]_{\mathtt{pln}} \left\{x \overset{\pi}{\mapsto} v * a = v\right\}$$

where x is a specification variable.

Release and acquire accesses

$$\begin{array}{l} \vdash \left\{x = e * \mathsf{Acq}(x, \phi) * \mathsf{O}(x, \_, t)\right\} \\ \quad a := [e]_{\mathtt{acq}} \\ \quad \left\{\exists t' \geq t.\ \mathsf{Acq}(x, \phi[a \mapsto \top]) * \phi(a) * \mathsf{O}(x, a, t')\right\} \end{array}$$

$$\begin{array}{l} \vdash \left\{x = e * \mathsf{Acq}(x, \phi) * \mathsf{W}^1(x, X)\right\} \\ \quad a := [e]_{\mathtt{acq}} \\ \quad \left\{\mathsf{Acq}(x, \phi[a \mapsto \top]) * \phi(a) * \mathsf{W}^1(x, X) * \mathsf{O}(x, a, \mathit{snd}(\max(X))) * a = \mathit{fst}(\max(X))\right\} \end{array}$$

$$\begin{array}{l} \vdash \left\{x = e_1 * v = e_2 * \mathsf{Rel}(x, \phi) * \mathsf{W}^\pi(x, X) * \phi(v)\right\} \\ \quad [e_1]_{\mathtt{rel}} := e_2 \\ \quad \left\{\exists t.\ \mathsf{W}^\pi(x, X \cup \{(v, t)\}) * \mathit{snd}(\max(X)) < t\right\} \end{array}$$

where x, t, v, π and X are specification variables.

Relaxed accesses

$$\begin{array}{l} \vdash \left\{x = e * \mathsf{Acq}(x, \phi) * \mathsf{O}(x, \_, t)\right\} \\ \quad a := [e]_{\mathtt{rlx}} \\ \quad \left\{\exists t' \geq t.\ \nabla(\phi(a)) * \mathsf{O}(x, a, t')\right\} \end{array}$$

$$\begin{array}{l} \vdash \left\{x = e * \mathsf{Acq}(x, \phi) * \mathsf{W}^1(x, X)\right\} \\ \quad a := [e]_{\mathtt{rlx}} \\ \quad \left\{\mathsf{Acq}(x, \phi) * \mathsf{W}^1(x, X) * (\nabla\phi(a)) * \mathsf{O}(x, a, \mathit{snd}(\max(X))) * a = \mathit{fst}(\max(X))\right\} \end{array}$$

$$\begin{array}{l} \vdash \left\{x = e_1 * v = e_2 * \mathsf{Rel}(x, \phi) * \mathsf{W}^\pi(x, X) * \phi(v) * \mathrm{pure}(\phi(v))\right\} \\ \quad [e_1]_{\mathtt{rlx}} := e_2 \\ \quad \left\{\exists t.\ \mathsf{W}^\pi(x, X \cup \{(v, t)\}) * \mathit{snd}(\max(X)) < t\right\} \end{array}$$

where x, t, v, π and X are specification variables.

3 Semantics of the program logic

3.1 Semantic domains

$$
\begin{aligned}
\mathit{PredId} &\stackrel{def}{=} \mathcal{N} \\
\mathit{Perm} &\stackrel{def}{=} \{x \in \mathbb{Q} \mid 0 \leq x \leq 1\} \\
\mathit{Perm}^{+} &\stackrel{def}{=} \{x \in \mathbb{Q} \mid 0 < x \leq 1\} \\
\mathit{PlnPerm} &\stackrel{def}{=} \mathit{PLoc} \to 1 + (\mathit{Perm}^{+} \times \mathit{Val} \times \mathit{Time}) \\
\mathit{WrPerm} &\stackrel{def}{=} \mathit{NPLoc} \to \{(\pi, X) \in \mathit{Perm} \times \mathcal{P}(\mathit{Val} \times \mathit{Time}) \mid \pi = 0 \Rightarrow X = \emptyset\} \\
\mathit{AcqPerm} &\stackrel{def}{=} \mathit{NPLoc} \to \mathcal{P}(\mathit{PredId}) \\
r \in \mathcal{M} &\stackrel{def}{=} \mathit{PlnPerm} \times \mathit{WrPerm} \times \mathit{AcqPerm} \\
u \in \mathit{MsgRes} &\stackrel{def}{=} \mathit{Msg} \rightharpoonup_{fin} (\mathit{PredId} \rightharpoonup_{fin} \mathcal{M}) \\
\mathcal{W} \in \mathit{World} &\stackrel{def}{=} (\mathit{PLoc} \to \mathit{Pred}) \times (\mathit{PredId} \rightharpoonup_{fin} \mathit{Pred}) \\
p \in \mathit{Prop} &\stackrel{def}{=} \mathit{World} \to_{mon} \mathcal{P}^{\uparrow}(\mathit{GConf} \times \mathit{View} \times \mathcal{M}) \\
\mathit{TRes} &\stackrel{def}{=} \mathit{TId} \to \mathcal{M}
\end{aligned}
$$

We use $r.\mathtt{pln}$, $r.\mathtt{wr}$ and $r.\mathtt{acq}$ to refer to the first, second and third projection of a resource r, respectively.

Resource monoid

$$
\begin{array}{|l|}
\hline
\bullet : \mathcal{M} \times \mathcal{M} \rightharpoonup \mathcal{M} \\
\varepsilon : \mathcal{M} \\
\hline
\end{array}
$$

The set of resources, $\mathcal{M}$ is a partial commutative monoid with the following composition operator and monoid unit.

$$
\begin{aligned}
r_1 \bullet r_2 &\stackrel{def}{=} (r_1.\mathtt{pln} \bullet_{\mathtt{pln}} r_2.\mathtt{pln}, r_1.\mathtt{wr} \bullet_{\mathtt{wr}} r_2.\mathtt{wr}, r_1.\mathtt{acq} \bullet_{\mathtt{acq}} r_2.\mathtt{acq}) \\
\varepsilon &\stackrel{def}{=} (\lambda\_.\ \mathit{inj}_1(*), \lambda\_.\ (0, \emptyset), \lambda\_.\ \emptyset)
\end{aligned}
$$

where

$$
h_1 \bullet_{\mathtt{pln}} h_2 \stackrel{def}{=} \begin{cases} \lambda x.\ h_1(x) \bullet_{pl} h_2(x) & \text{if } h_1(x) \# h_2(x) \text{ for all } x \in \mathit{Loc} \\ \mathit{undefined} & \text{otherwise} \end{cases}
$$

$$
\mathit{inj}_1(*) \bullet_{pl} x \stackrel{def}{=} x
$$

$$
x \bullet_{pl} \mathit{inj}_1(*) \stackrel{def}{=} x
$$

$$
\mathit{inj}_2(\pi_1, v_1) \bullet_{pl} \mathit{inj}_2(\pi_2, v_2) \stackrel{def}{=} \begin{cases} \mathit{inj}_2(\pi_1 + \pi_2, v_1) & \text{if } \pi_1 + \pi_2 \leq 0 \text{ and } v_1 = v_2 \\ \bot & \text{otherwise} \end{cases}
$$

$$
f_1 \bullet_{\mathtt{wr}} f_2 \stackrel{def}{=} \begin{cases} \lambda x.\ f_1(x) \bullet f_2(x) & \text{if } f_1(x) \# f_2(x) \text{ for all } x \\ \mathit{undefined} & \text{otherwise} \end{cases}
$$

$$
(\pi_1, X_1) \bullet (\pi_2, X_2) \stackrel{def}{=} \begin{cases} (\pi_1 + \pi_2, X_1 \cup X_2) & \text{if } \pi_1 + \pi_2 \leq 0 \\ \mathit{undefined} & \text{otherwise} \end{cases}
$$

$$
g_1 \bullet_{\mathtt{acq}} g_2 \stackrel{def}{=} \begin{cases} \lambda x.\ g_1(x) \cup g_2(x) & \text{if } \forall x.\ g_1(x) \cap g_2(x) = \emptyset \\ \mathit{undefined} & \text{otherwise} \end{cases}
$$

The monotonicity of propositions is with respect to the following ordering on worlds, and subset inclusion on $\mathcal{P}^{\uparrow}(\mathit{GConf} \times \mathit{View} \times \mathcal{M})$.

World ordering $\boxed{\leq : \mathcal{P}(\mathit{World} \times \mathit{World})}$

$$\mathcal{W}_1 \leq \mathcal{W}_2 \stackrel{def}{=} \mathcal{W}_1.\mathtt{rel} = \mathcal{W}_2.\mathtt{rel} \wedge \mathit{dom}(\mathcal{W}_1.\mathtt{acq}) \subseteq \mathit{dom}(\mathcal{W}_2.\mathtt{acq}) \wedge \\ \forall \iota \in \mathit{dom}(\mathcal{W}_1.\mathtt{acq}).\ \mathcal{W}_1.\mathtt{acq}(\iota) = \mathcal{W}_2.\mathtt{acq}(\iota)$$

The upwards-closure on $\mathit{GConf} \times \mathit{View} \times \mathcal{M}$ is with respect to the point-wise extension of the following orders on resources and global configurations and the previously defined ordering on views.

Resource ordering $\boxed{\leq : \mathcal{P}(\mathcal{M} \times \mathcal{M})}$

$$r_1 \leq r_2 \stackrel{def}{=} \exists r_3.\ r_2 = r_1 \bullet r_3$$

Configuration ordering $\boxed{\leq : \mathcal{P}(\mathit{GConf} \times \mathit{GConf})}$

$$\mathit{gconf}_1 \leq \mathit{gconf}_2 \stackrel{def}{=} \mathit{gconf}'.M \text{ is a future memory of } \mathit{gconf}.M \text{ w.r.t. } \mathit{gconf}'.P$$

Lemma 2.

$$\mathit{gconf}.M \text{ is closed} \wedge (\Sigma, \mathit{gconf}) \Longrightarrow (\Sigma', \mathit{gconf}') \Rightarrow \mathit{gconf} \leq \mathit{gconf}'$$

Well-formed configuration $\boxed{\mathrm{wf}_{\mathrm{conf}}}$

$$\mathrm{wf}_{\mathrm{conf}}(i, \mathit{gconf}, V) \stackrel{def}{=} \forall m \in \mathit{gconf}.P.\ m.\mathtt{tid} = i \Rightarrow V.\mathtt{rlx}(m.\mathtt{loc}) < m.\mathtt{time}$$

Lemma 3.

$$\langle\langle \sigma, V\rangle, \langle M, P\rangle\rangle \xrightarrow{\mathrm{NP}}{}_i^{\ *} \langle\langle \sigma', V'\rangle, \langle M', P'\rangle\rangle \wedge P'(i) = \emptyset \Rightarrow \mathrm{wf}_{\mathrm{conf}}(i, \langle M, P\rangle, V)$$

Read assertion $\boxed{O}$

$$O(\ell, v, t) \stackrel{def}{=} \lambda\mathcal{W}.\ \{(\mathit{gconf}, V, r) \mid \exists m \in \mathit{gconf}.M.\ m.\mathtt{loc} = \ell \wedge m.\mathtt{val} = v \wedge m.\mathtt{time} = t \wedge t \leq V.\mathtt{rlx}(\ell)\}$$

Write assertion $\boxed{W}$

$$W_\pi(\ell, X) \stackrel{def}{=} \lambda\mathcal{W}.\ \{(\mathit{gconf}, V, r) \mid \exists \pi' \geq \pi.\ r.\mathtt{wr}(\ell) = \mathit{inj}_2(\pi', X) \wedge \mathit{snd}(\max(X)) \leq V.\mathtt{rlx}(\ell)\}$$

Acq assertion $\boxed{\mathit{acq}}$

$$\mathit{acq}(\ell, \phi) \stackrel{def}{=} \lambda\mathcal{W}.\ \{(\mathit{gconf}, V, r) \mid \exists \iota \in \mathit{dom}(\mathcal{W}.\mathtt{acq}) \cap r.\mathtt{acq}(\ell).\ \mathcal{W}.\mathtt{acq}(\iota) = \phi\}$$

Rel assertion $\boxed{\mathit{rel}}$

$$\mathit{rel}(\ell, \phi) \stackrel{def}{=} \lambda\mathcal{W}.\ \{(\mathit{gconf}, V, r) \mid \vdash \forall v.\ \phi(v) \Rightarrow \mathcal{W}.\mathtt{rel}(\ell)(v)\}$$

Points-to assertion $\boxed{\ell \overset{\pi}{\mapsto} v}$

$$\ell \overset{\pi}{\mapsto} v \overset{def}{=} \lambda \mathcal{W}.\ \{(gconf, V, r) \mid \exists \pi' \geq \pi.\ \exists t.\ r.\mathtt{pln}(\ell) = inj_2(\pi', v, t) \wedge t \leq V.\mathtt{pln}(x)\}$$

Assertion interpretation $\boxed{[\![-]\!]^{\equiv}_{=} : Assn \times TStore \times Env \to Prop}$

$$\begin{aligned}
[\![\bot]\!]^{\eta}_{\mu}(\mathcal{W}) &\overset{def}{=} \emptyset \\
[\![\top]\!]^{\eta}_{\mu}(\mathcal{W}) &\overset{def}{=} GConf \times View \times \mathcal{M} \\
[\![P \wedge Q]\!]^{\eta}_{\mu}(\mathcal{W}) &\overset{def}{=} [\![P]\!]^{\eta}_{\mu} \cap [\![Q]\!]^{\eta}_{\mu}(\mathcal{W}) \\
[\![P \vee Q]\!]^{\eta}_{\mu}(\mathcal{W}) &\overset{def}{=} [\![P]\!]^{\eta}_{\mu}(\mathcal{W}) \cup [\![Q]\!]^{\eta}_{\mu}(\mathcal{W}) \\
[\![P \Rightarrow Q]\!]^{\eta}_{\mu}(\mathcal{W}) &\overset{def}{=} \{(gconf, V, r) \mid \forall gconf' \geq gconf.\ \forall V' \geq V.\ \forall r' \geq r.\ \forall \mathcal{W}' \geq \mathcal{W}. \\
&\qquad (gconf', V', r') \in [\![P]\!]^{\eta}_{\mu}(\mathcal{W}') \Rightarrow (gconf', V', r') \in [\![Q]\!]^{\eta}_{\mu}(\mathcal{W}')\} \\
[\![P * Q]\!]^{\eta}_{\mu}(\mathcal{W}) &\overset{def}{=} \{(gconf, V, r_1 \bullet r_2) \mid (gconf, V, r_1) \in [\![P]\!]^{\eta}_{\mu}(\mathcal{W}) \wedge (gconf, V, r_2) \in [\![Q]\!]^{\eta}_{\mu}(\mathcal{W})\} \\
[\![\forall x.\ P]\!]^{\eta}_{\mu}(\mathcal{W}) &\overset{def}{=} \textstyle\bigcap_{v \in Val} [\![P]\!]^{\eta[x \mapsto v]}_{\mu}(\mathcal{W}) \\
[\![\exists x.\ P]\!]^{\eta}_{\mu}(\mathcal{W}) &\overset{def}{=} \textstyle\bigcup_{v \in Val} [\![P]\!]^{\eta[x \mapsto v]}_{\mu}(\mathcal{W}) \\
[\![N_1 = N_2]\!]^{\eta}_{\mu}(\mathcal{W}) &\overset{def}{=} \{x \mid [\![N_1]\!]^{\eta}_{\mu} = [\![N_2]\!]^{\eta}_{\mu}\} \\
[\![(\lambda x.\ P)(N)]\!]^{\eta}_{\mu}(\mathcal{W}) &\overset{def}{=} [\![P]\!]^{\eta[x \mapsto [\![N]\!]^{\eta}_{\mu}]}_{\mu}(\mathcal{W})
\end{aligned}$$

$$\begin{aligned}
[\![\mathsf{O}(\ell, v, t)]\!]^{\eta}_{\mu}(\mathcal{W}) &\overset{def}{=} O([\![\ell]\!]^{\eta}_{\mu}, [\![v]\!]^{\eta}_{\mu}, [\![t]\!]^{\eta}_{\mu})(\mathcal{W}) \\
[\![\mathsf{W}_{\pi}(\ell, X)]\!]^{\eta}_{\mu}(\mathcal{W}) &\overset{def}{=} W_{[\![\pi]\!]^{\eta}_{\mu}}([\![\ell]\!]^{\eta}_{\mu}, [\![X]\!]^{\eta}_{\mu})(\mathcal{W}) \\
[\![\mathsf{Acq}(\ell, \phi)]\!]^{\eta}_{\mu}(\mathcal{W}) &\overset{def}{=} acq([\![\ell]\!]^{\eta}_{\mu}, \phi)(\mathcal{W}) \\
[\![\mathsf{Rel}(\ell, \phi)]\!]^{\eta}_{\mu}(\mathcal{W}) &\overset{def}{=} rel([\![\ell]\!]^{\eta}_{\mu}, \phi)(\mathcal{W}) \\
\left[\!\!\left[\ell \overset{\pi}{\mapsto} v\right]\!\!\right]^{\eta}_{\mu}(\mathcal{W}) &\overset{def}{=} ([\![\ell]\!]^{\eta}_{\mu} \overset{[\![\pi]\!]^{\eta}_{\mu}}{\mapsto} [\![v]\!]^{\eta}_{\mu})(\mathcal{W}) \\
[\![\nabla P]\!]^{\eta}_{\mu}(\mathcal{W}) &\overset{def}{=} \{(gconf, V, r) \mid [\![P]\!]^{\eta}_{\mu}(\mathcal{W}) \neq \emptyset\} \\
[\![\mathrm{pure}(P)]\!]^{\eta}_{\mu}(\mathcal{W}) &\overset{def}{=} \{x \mid \forall \mathcal{W}' \geq \mathcal{W}.\ \forall x_1, x_2.\ x_1 \in [\![P]\!]^{\eta}_{\mu}(\mathcal{W}') \Rightarrow x_2 \in [\![P]\!]^{\eta}_{\mu}(\mathcal{W}')\}
\end{aligned}$$

3.2 Acquirable resources

Acquirable messages $\boxed{\text{canAcq} : \mathit{AcqPerm} \times \mathit{MsgRes} \to \mathcal{P}(\mathit{Msg} \times \mathit{PredId})}$

The $\text{canAcq}(A, u)$ function recursively computes the set of messages whose resources we are currently allowed to acquire from message resource assignment u.

$$\text{canAcq}(A, u) \stackrel{def}{=} \begin{cases} I \cup \text{canAcq}(\Pi_{(m,\iota)\in I} u(m)(\iota).\mathtt{acq}, u[I \mapsto \bot]) & \text{if } I = \text{canAcq}_0(A, u) \neq \emptyset \\ \emptyset & \text{otherwise} \end{cases}$$

$$\text{canAcq}_0(A, u) \stackrel{def}{=} \{(m, \iota) \mid m \in \mathit{dom}(u) \wedge \iota \in A(m.\mathtt{loc}) \cap \mathit{dom}(u(m))\}$$

The $\text{canAcq}(A, u)$ function is defined by recursion using $\Sigma_{m\in dom(u)}|dom(u(m))|$ as the size function.

Acquirable resources $\boxed{\text{canAcq} : \mathcal{M} \times \mathit{MsgRes} \to \mathcal{M}}$

The $\text{canAcq}(r, u)$ function computes the composition of the resources we are currently allowed to acquire from message resource assignment u.

$$\text{canAcq}(r, u) \stackrel{def}{=} \Pi_{(m,\iota)\in\text{canAcq}(r.\mathtt{acq},u)} u(m)(\iota)$$

Throughout the rest of the document whenever we write $\text{canAcq}(r, u)$ we will implicitly assume that $r \bullet \Pi_{m\in dom(u)}\Pi_{\iota\in dom(u(m))} u(m)(\iota)$ is defined and whenever we write $\text{canAcq}(A, u)$ that $A\bullet(\Pi_{m\in dom(u)}\Pi_{\iota\in dom(u(m))} u(m)(\iota)).\mathtt{acq}$ is defined.

Lemma 4. $\text{canAcq}_0(A \bullet B, u) = \text{canAcq}_0(A, u) \cup \text{canAcq}(B, u)$

Lemma 5. $\text{canAcq}(A \bullet B, u) = \text{canAcq}(A, u) \cup \text{canAcq}(B, u)$

Lemma 6. For $m \in \mathit{dom}(u)$ and $\iota \in A(m.\mathtt{loc}) \cap \mathit{dom}(u(m))$, we have $\text{canAcq}(A, u) = \text{canAcq}(A', u')$ where $A' = A \bullet u(m)(\iota).A$ and $u' = u[m, \iota \mapsto \varepsilon]$.

Proof.

$$\begin{aligned} \text{canAcq}(A', u') &= \text{canAcq}(A, u') \cup \text{canAcq}(u(m)(\iota).A, u') \\ &= (\text{canAcq}(A, u) \setminus \text{canAcq}(u(m)(\iota).A, u)) \cup \text{canAcq}(u(m)(\iota).A, u) \\ &= \text{canAcq}(A, u) \end{aligned}$$

□

Lemma 7.

$$\begin{aligned} &m \in \mathit{dom}(u) \wedge \iota \in r.\mathtt{acq}(m.\mathtt{loc}) \cap \mathit{dom}(u(m)) \Rightarrow \\ &\quad \text{canAcq}(r, u) = u(m)(\iota) \bullet \text{canAcq}(r \bullet u(m)(\iota), u[m, \iota \mapsto \varepsilon]) \end{aligned}$$

Proof. By Lemma 6

$$\begin{aligned} &u(m)(\iota) \bullet \text{canAcq}(r \bullet u(m)(\iota), u[m, \iota \mapsto \varepsilon]) \\ &= u(m)(\iota) \bullet \Pi_{(m',\iota')\in\text{canAcq}((r\bullet u(m)(\iota)).\mathtt{acq}, u[m,\iota\mapsto\varepsilon])} u[m, \iota \mapsto \varepsilon](m')(\iota') \\ &= u(m)(\iota) \bullet \Pi_{(m',\iota')\in\text{canAcq}(r.\mathtt{acq},u)} u[m, \iota \mapsto \varepsilon](m')(\iota') \\ &= u(m)(\iota) \bullet \Pi_{(m',\iota')\in(\text{canAcq}(r.\mathtt{acq},u)\setminus\{(m,\iota)\})} u(m')(\iota') \\ &= \Pi_{(m',\iota')\in\text{canAcq}(r.\mathtt{acq},u)} u(m')(\iota') \\ &= \text{canAcq}(r, u) \end{aligned}$$

□

Lemma 8.

$$m \in dom(u) \wedge \iota \in A(m.\texttt{loc}) \cap dom(u(m)) \Rightarrow \\ \text{canAcq}(A, u) \setminus \{(m, \iota)\} = \text{canAcq}(A', u')$$

where $A' = A \bullet u(m)(\iota).\texttt{acq}$ and $u' = u[m, \iota \mapsto \bot]$.

Proof.

$$\begin{aligned}\text{canAcq}(A', u') &= \text{canAcq}(A, u') \cup \text{canAcq}(u(m)(\iota).\texttt{acq}, u') \\ &= (\text{canAcq}(A, u) \setminus (\{(m, \iota)\} \cup \text{canAcq}(u(m)(\iota).\texttt{acq}, u))) \cup \text{canAcq}(u(m)(\iota).\texttt{acq}, u) \\ &= \text{canAcq}(A, u) \setminus \{(m, \iota)\}\end{aligned}$$

□

Lemma 9.

$$\iota \in A(x) \wedge \iota' \text{ fresh for } A \text{ and } u \Rightarrow \\ \text{canAcq}_0(A', u') = (\text{canAcq}_0(A, u) \setminus \{(m, \iota) \mid m.\texttt{loc} = x\}) \cup \\ \{(m, \iota') \mid m.\texttt{loc} = x \wedge \iota \in dom(u(m))\}$$

where $u' = u[\iota \rightsquigarrow_x \iota']$ and $A' = A[x \mapsto A(x)[\iota \rightsquigarrow \iota']]$.

Lemma 10.

$$\iota \in A(x) \wedge \iota' \text{ fresh for } A \text{ and } u \Rightarrow \\ \text{canAcq}(A', u') = (\text{canAcq}(A, u) \setminus \{(m, \iota) \mid m.\texttt{loc} = x\}) \cup \\ \{(m, \iota') \mid m.\texttt{loc} = x \wedge \iota \in dom(u(m))\}$$

where $u' = u[\iota \rightsquigarrow_x \iota']$ and $A' = A[x \mapsto A(x)[\iota \rightsquigarrow \iota']]$.

Lemma 11.

$$\iota \notin r.\texttt{acq}(x) \wedge \iota' \text{ fresh for } u \text{ and } r \wedge \\ (\forall m \in dom(u).\ \forall \iota'' \in dom(u(m)).\ \iota \notin u(m)(\iota'').A(x)) \Rightarrow \\ \text{canAcq}(r[\iota \rightsquigarrow_x \iota'], u[\iota \rightsquigarrow_x \iota']) = \text{canAcq}(r, u)$$

Proof. It follows from the $\iota \notin r.\texttt{acq}(x)$ assumption that $r = r[\iota \rightsquigarrow_x \iota']$. Furthermore, since $\iota \notin u(m)(\iota'').A(x)$ for any $m \in dom(m)$ and $\iota'' \in dom(u(m))$ it follows that the acquirable resources should be independent of the current resources associated with the predicate named ι, since these resources currently cannot be acquired. □

Predicate renaming

$$\boxed{\begin{array}{l}(-)[=\rightsquigarrow_x\equiv] : \mathit{MsgRes} \times \mathit{PredId} \times \mathit{PredId} \to \mathit{MsgRes} \\ (-)[=\rightsquigarrow_x\equiv] : \mathcal{M} \times \mathit{PredId} \times \mathit{PredId} \to \mathcal{M}\end{array}}$$

$$u[\iota \rightsquigarrow_x \iota'] \stackrel{def}{=} \lambda m.\ \begin{cases} \lambda \iota''.\ \begin{cases} u(m)(\iota) & \text{if } \iota'' = \iota' \text{ and } m.\texttt{loc} = x \\ \bot & \text{if } \iota'' = \iota \text{ and } m.\texttt{loc} = x \\ u(m)(\iota'') & \text{otherwise} \end{cases} & \text{if } m \in dom(u) \\ \bot & \text{otherwise} \end{cases}$$

$$X[\iota \rightsquigarrow \iota'] \stackrel{def}{=} \mathit{if}\ \iota \in X\ \mathit{then}\ (X \setminus \{\iota\}) \cup \{\iota'\}\ \mathit{else}\ X$$

$$r[\iota \rightsquigarrow_x \iota'] \stackrel{def}{=} r[A(x) \mapsto A(x)[\iota \rightsquigarrow_x \iota']]$$

$$r_F[\iota \rightsquigarrow_x \iota'] \stackrel{def}{=} \lambda t \in \mathit{TId}.\ r_F(t)[\iota \rightsquigarrow_x \iota']$$

Lemma 12.

$$\begin{aligned}&\iota \in A(x) \wedge \iota' \text{ fresh for } A \text{ and } u \Rightarrow \\ &\quad \text{canAcq}(A', u') = (\text{canAcq}(A, u) \setminus \{(m, \iota) \mid m \in dom(u) \wedge m.\texttt{loc} = x\}) \cup \\ &\qquad\qquad \{(m, \iota') \mid (m, \iota) \in \text{canAcq}(A, u) \wedge m.\texttt{loc} = x\}\end{aligned}$$

where $u' = u[\iota \rightsquigarrow_x \iota']$ and $A' = A[x \mapsto A(x)[\iota \rightsquigarrow \iota']]$.

Lemma 13.

$$\begin{aligned}&\iota \in r.\texttt{acq}(x) \wedge \iota' \text{ fresh for } u \text{ and } r \Rightarrow \\ &\quad \text{canAcq}(r, u) = \text{canAcq}(r[\iota \rightsquigarrow_x \iota'], u[\iota \rightsquigarrow_x \iota'])\end{aligned}$$

Proof. Follows from Lemma 12. □

Lemma 14.

$$\begin{aligned}&dom(u) = dom(u') = M(\texttt{rel}) \wedge x \in dom(r.\texttt{acq}) \wedge \iota \in r.\texttt{acq}(x) \wedge \\ &r' = r[A(x) \mapsto (r.\texttt{acq}(x) \setminus \{\iota\}) \cup \{\iota_1, \iota_2\}] \wedge \\ &u' = u[m \in M(\texttt{rel}, x) \mapsto u(m)[\iota \mapsto \bot, \iota_1 \mapsto r^1_m, \iota_2 \mapsto r^2_m]] \\ &\forall m \in M(\texttt{rel}, x).\ u(m)(\iota) = r^1_m \bullet r^2_m \\ &\quad \iota \in dom(u(m)) \wedge \iota_1, \iota_2 \notin dom(u(m)) \cup dom(u'(m)) \wedge \\ &\Rightarrow \text{canAcq}(r, u) = \text{canAcq}(r', u')\end{aligned}$$

Write restrictions

$$\boxed{\begin{array}{l}\text{writeAllowed} : \mathcal{P}(\mathcal{M} \times \mathit{Msg}) \\ \text{writesAllowed} : \mathcal{P}(\mathcal{M} \times \mathcal{P}(\mathit{Msg}))\end{array}}$$

$$\begin{aligned}\text{writeAllowed}(r, m) &\stackrel{def}{=} (\mathit{pln}(m) \wedge r.\texttt{pln}(m.\texttt{loc}) = (1, \_)) \vee \\ &\quad (\mathit{npln}(m) \wedge \pi_1(r.\texttt{wr}(m.\texttt{loc})) > 0) \\ \text{writesAllowed}(r, M) &\stackrel{def}{=} \forall m \in M.\ \text{writeAllowed}(r, m)\end{aligned}$$

Ownership ordering

$$\boxed{\leq_o : \mathcal{P}(\mathcal{M} \times \mathcal{M})}$$

$$\begin{aligned}r_1 \leq_{\text{o}} r_2 \stackrel{def}{=}\ & \forall x.\ \pi_1(r_1.\texttt{wr}(x)) \leq \pi_1(r_2.\texttt{wr}(x)) \wedge \\ & \forall x.\ r_2.\texttt{pln}(x) = \mathit{inj}_1(*) \Rightarrow r_1.\texttt{pln}(x) = \mathit{inj}_1(*) \wedge \\ & \forall x.\ r_1.\texttt{pln}(x) \neq \mathit{inj}_1(*) \Rightarrow \pi_1(r_1.\texttt{pln}(x)) \leq \pi_1(r_2.\texttt{pln}(x))\end{aligned}$$

Lemma 15.

$$r_1 \leq_{\text{o}} r_1' \wedge r_2 \leq_{\text{o}} r_2' \Rightarrow r_1 \bullet r_2 \leq_{\text{o}} r_1' \bullet r_2'$$

Lemma 16.

$$\text{writeAllowed}(r_1, m) \wedge r_1 \leq_{\text{o}} r_2 \Rightarrow \text{writeAllowed}(r_2, m)$$

Lemma 17.

$$\begin{aligned}&\iota \notin \mathit{dom}(u(m)) \wedge \mathit{def}(r, u[m \mapsto u(m)[\iota \mapsto r_2]]) \\ &\Rightarrow \text{canAcq}(r_1, u[m \mapsto u(m)[\iota \mapsto r_2]]) \leq_{\text{o}} \text{canAcq}(r_1 \bullet r_2, u)\end{aligned}$$

World erasure

$$\boxed{|-| : \mathit{World} \to \mathit{World}}$$

$$|\mathcal{W}| \stackrel{def}{=} (\pi_1(\mathcal{W}), [])$$

Erasure

$$\boxed{\lfloor =_1, =_2, =_3 \rfloor_{=_4} : \mathcal{M} \times \mathit{MsgRes} \times \mathit{World} \times \mathcal{P}(\mathit{TId}) \to \mathcal{P}(\mathit{GConf})}$$

$$\begin{aligned}\lfloor r, u, \mathcal{W} \rfloor_T \stackrel{def}{=} \{&(M, P) \mid \mathit{closed}(M) \wedge \\ &\mathit{let}\ t = \Pi_{t \in \mathit{TId}} r(t) \bullet \Pi_{m \in M} \Pi_{\iota \in \mathit{dom}(u(m))} u(m)(\iota)\ \mathit{in} \\ &\quad \text{wf}(\mathcal{W}, u, t.A) \wedge \\ &\quad \forall x \in \mathit{locs}(\mathit{pln}(M)).\ t.\texttt{pln}(x) \neq \mathit{inj}_1(*) \Rightarrow \\ &\qquad \mathit{last}(M(x) \setminus P).\texttt{val} = t.\texttt{pln}(x).\texttt{val} \wedge \\ &\qquad \mathit{last}(M(x) \setminus P).\texttt{time} = t.\texttt{pln}(x).\texttt{time} \wedge \\ &\quad \forall x \in \mathit{locs}(\mathit{npln}(M)). \\ &\qquad \{(m.\texttt{val}, m.\texttt{time}) \mid m \in M(x) \setminus P\} = \mathit{snd}(t.\texttt{wr}(x)) \wedge \\ &\quad \forall M(\texttt{rlx}). \\ &\qquad (\mathit{gconf}_\perp, V_\perp, \varepsilon) \in [\![\mathcal{W}.\texttt{rel}(m.\texttt{loc})(m.\texttt{val})]\!]^{[]}_{[]}(|\mathcal{W}|) \wedge \\ &\quad \forall m \in M(\texttt{rel}).\ t.\texttt{acq}(m.\texttt{loc}) = \mathit{dom}(u(m)) \wedge \forall \iota \in \mathit{dom}(u(m)). \\ &\qquad (\mathit{gconf}, m.\texttt{view}, u(m)(\iota)) \in [\![\mathcal{W}.\texttt{acq}(\iota)(m.\texttt{val})]\!]^{[]}_{[]}(\mathcal{W}) \wedge \\ &\quad \text{validPromises}_T(r, u, P)\}\end{aligned}$$

where

$$\begin{aligned}\mathrm{wf}(\mathcal{W}, u, A) \stackrel{def}{=}\ & dom(u) = M(\mathtt{rel}) \wedge \\ & \{\iota \mid m \in dom(u) \wedge \iota \in dom(u(m))\} \subseteq dom(\mathcal{W}.\mathtt{acq}) \wedge \\ & \forall x \in NPLoc.\ \forall v.\ \vdash \mathcal{W}.\mathtt{rel}(x)(v) \Rightarrow \circledast_{\iota \in A(x)} \mathcal{W}.\mathtt{acq}(\iota)(v)\end{aligned}$$

Valid promises

$$\boxed{\begin{array}{l}\mathrm{validPromises}_{\_}(=) : \mathcal{P}(\mathcal{P}(\mathit{TId}) \times \mathit{TRes} \times \mathit{MsgRes} \times \mathit{Mem}) \\ \quad \mathrm{validPromise}(-) : \mathcal{P}(\mathcal{M} \times \mathit{MsgRes} \times \mathit{Msg})\end{array}}$$

$$\begin{aligned}\mathrm{validPromises}_T(r, u, P) &\stackrel{def}{=} \forall m \in P.\ m.\mathtt{tid} \notin T \Rightarrow \mathrm{validPromise}(r, u, m) \\ \mathrm{validPromise}(r, u, m) &\stackrel{def}{=} \mathrm{writeAllowed}(r(m.\mathtt{tid}) \bullet \mathrm{canAcq}(r(m.\mathtt{tid}), u), m)\end{aligned}$$

Non-promising safety assertion

$$\boxed{\mathrm{npsafe}_{\_}(=) : \omega^2 \times \mathit{TLState} \times \mathit{Prop} \to \mathit{Prop}}$$

$$\begin{aligned}&\mathrm{npsafe}_{(0,m)}(\sigma, B)(\mathcal{W}) \stackrel{def}{=} \mathit{GConf} \times \mathit{View} \times \mathcal{M} \\ &\mathrm{npsafe}_{(n+1,0)}(\sigma, B)(\mathcal{W}) \stackrel{def}{=} \{(\mathit{gconf}, V, r) \mid \forall m.\ (\mathit{gconf}, V, r) \in \mathrm{npsafe}_{(n,m)}(\sigma, B)(\mathcal{W})\} \\ &\mathrm{npsafe}_{(n+1,m+1)}(\sigma, B)(\mathcal{W}) \stackrel{def}{=} \{(\mathit{gconf}_1, V_1, r_1) \mid \forall (\mathit{gconf}, V, r) \geq (\mathit{gconf}_1, V_1, r_1). \\ &\quad (\sigma.s = \mathbf{skip} \Rightarrow (\mathit{gconf}, V, r) \in vs(B(\sigma.\mu))(\mathcal{W})) \\ &\quad \wedge (\forall r_F, f, \sigma', \mathit{gconf}', V', u, i, \mathcal{W}' \geq \mathcal{W}.\ \mathit{gconf} \in \lfloor r_F[i \mapsto r \bullet f], u, \mathcal{W}' \rfloor_{\{i\}} \wedge \\ &\qquad \langle\langle \sigma, V\rangle, \mathit{gconf}\rangle \xrightarrow{\mathrm{NP}}_i \langle\langle \sigma', V'\rangle, \mathit{gconf}'\rangle \wedge \mathrm{wf}_{\mathrm{conf}}(i, \mathit{gconf}, V) \wedge \mathrm{wf}_{\mathrm{conf}}(i, \mathit{gconf}', V') \Rightarrow \\ &\quad \exists r', u', \mathcal{W}'' \geq \mathcal{W}'.\ \mathit{gconf}' \in \lfloor r_F[i \mapsto r' \bullet f], u', \mathcal{W}'' \rfloor_{\{i\}} \wedge \\ &\qquad (\mathit{gconf}', V', r') \in \mathrm{npsafe}_{(n+1,m)}(\sigma', B)(\mathcal{W}'') \wedge \\ &\qquad r' \bullet \mathrm{canAcq}(r', u') \leq_{\mathrm{o}} r \bullet \mathrm{canAcq}(r, u) \wedge \\ &\qquad \mathrm{writesAllowed}(r, \mathrm{written}(\mathit{gconf}, \mathit{gconf}'))\}\end{aligned}$$

Lemma 18 (npsafe is downwards-closed.)**.**

$$\begin{aligned}&\forall n, n', m, m', \sigma, B, \mathcal{W}. \\ &\quad (n', m') \leq (n, m) \wedge \mathrm{npsafe}_{(n,m)}(\sigma, B)(\mathcal{W}) \subseteq \mathrm{npsafe}_{(n',m')}(\sigma, B)(\mathcal{W})\end{aligned}$$

Written messages

$$\boxed{\mathrm{written} : \mathit{GConf} \times \mathit{GConf} \to \mathit{Mem}}$$

$$\mathrm{written}(\mathit{gconf}, \mathit{gconf}') \stackrel{def}{=} \{m \in \mathit{gconf}'.M \mid (m \notin \mathit{gconf}.M \vee m \in \mathit{gconf}.P) \wedge m \notin \mathit{gconf}'.P\}$$

View-shift assertion

$$\boxed{vs(-) : \mathit{Prop} \to \mathit{Prop}}$$

$$\begin{aligned}vs(B)(\mathcal{W}) \stackrel{def}{=} \{&(\mathit{gconf}_1, V_1, r_1) \mid \forall (\mathit{gconf}, V, r) \geq (\mathit{gconf}_1, V_1, r_1).\ \forall i, f, u, T, \mathcal{W}' \geq \mathcal{W}. \\ &\quad \mathit{gconf} \in \lfloor r_F[i \mapsto r \bullet f], u, \mathcal{W}' \rfloor_T \wedge \mathrm{wf}_{\mathrm{conf}}(i, \mathit{gconf}, V) \Rightarrow \\ &\quad\ \exists r', u', \mathcal{W}'' \geq \mathcal{W}'. \\ &\quad\quad (\mathit{gconf}, V, r') \in B(\mathcal{W}'') \wedge \mathit{gconf} \in \lfloor r_F[i \mapsto r' \bullet f], u', \mathcal{W}'' \rfloor_T \wedge \\ &\quad\quad r' \leq_{\mathrm{o}} r \wedge r' \bullet \mathrm{canAcq}(r', u') \leq_{\mathrm{o}} r \bullet \mathrm{canAcq}(r, u)\}\end{aligned}$$

Promising safety assertion $\boxed{\text{safe}_{\_}(=) : \omega \times \mathit{TLState} \times \mathit{Prop} \to \mathit{Prop}}$

$$
\begin{aligned}
&\text{safe}_0(\sigma, B)(\mathcal{W}) \stackrel{def}{=} \mathit{GConf} \times \mathit{View} \times \mathcal{M} \\
&\text{safe}_{n+1}(\sigma, B)(\mathcal{W}) \stackrel{def}{=} \{(\mathit{gconf}_1, V_1, r_1) \mid \forall (\mathit{gconf}, V, r) \geq (\mathit{gconf}_1, V_1, r_1). \\
&\quad (\sigma.s = \textsf{skip} \Rightarrow (\mathit{gconf}, V, r) \in \mathit{vs}(B(\sigma.\mu))(\mathcal{W})) \\
&\quad \wedge (\forall r_F, \sigma', \mathit{gconf}', V', u, \mathcal{W}', i.\ \mathit{gconf}' \in \lfloor r_F[i \mapsto r], u, \mathcal{W} \rfloor_\emptyset \wedge \\
&\qquad \langle\langle \sigma, V\rangle, \mathit{gconf}\rangle \Longrightarrow_i \langle\langle \sigma', V'\rangle, \mathit{gconf}'\rangle \wedge n > 0 \\
&\qquad \Rightarrow \exists r', u', \mathcal{W}'' \geq \mathcal{W}'. \\
&\qquad\quad \mathit{gconf}' \in \lfloor r_F[i \mapsto r'], u', \mathcal{W}'' \rfloor_\emptyset \wedge \\
&\qquad\quad (\mathit{gconf}', V', r') \in \text{safe}_n(\sigma', B)(\mathcal{W}''))\}
\end{aligned}
$$

Lemma 19 (safe is downwards-closed.)**.**

$$
\begin{aligned}
&\forall n, n', \sigma, B, \mathcal{W}. \\
&\quad n' \leq n \wedge \text{safe}_n(\sigma, B)(\mathcal{W}) \subseteq \text{safe}_{n'}(\sigma, B)(\mathcal{W})
\end{aligned}
$$

Interpretation of triples $\boxed{\models_{np} \{A\}\, s\, \{B\}, \models_p \{A\}\, s\, \{B\}, \models A \Rrightarrow B}$

$$
\begin{aligned}
\models_{np} \{A\}\, s\, \{B\} &\triangleq \forall n, m, \mu, \mathcal{W}.\ A(\mu)(\mathcal{W}) \subseteq \text{npsafe}_{(n,m)}((\mu, s), B)(\mathcal{W}) \\
\models_p \{A\}\, s\, \{B\} &\triangleq \forall n, \mu, \mathcal{W}.\ A(\mu)(\mathcal{W}) \subseteq \text{safe}_n((\mu, s), B)(\mathcal{W}). \\
\models A \Rrightarrow B &\stackrel{def}{=} \forall \mu, \mathcal{W}.\ A(\mu)(\mathcal{W}) \subseteq \mathit{vs}(B(\mu))(\mathcal{W})
\end{aligned}
$$

$$
\begin{aligned}
[\![\vdash \{P\}\ s\ \{Q\}]\!] &\stackrel{def}{=} \forall \eta.\ \models_p \{\lambda\mu.\ \lambda\mathcal{W}.\ [\![P]\!]^\eta_\mu(\mathcal{W})\}\, s\, \{\lambda\mu.\ \lambda\mathcal{W}.\ [\![Q]\!]^\eta_\mu(\mathcal{W})\} \\
[\![\vdash \{P\}\ s\ \{Q\}]\!]^{\text{np}} &\stackrel{def}{=} \forall \eta.\ \models_{np} \{\lambda\mu.\ \lambda\mathcal{W}.\ [\![P]\!]^\eta_\mu(\mathcal{W})\}\, s\, \{\lambda\mu.\ \lambda\mathcal{W}.\ [\![Q]\!]^\eta_\mu(\mathcal{W})\} \\
[\![\vdash P \Rrightarrow Q]\!] &\stackrel{def}{=} \forall \eta.\ \models (\lambda\mu.\ \lambda\mathcal{W}.\ [\![P]\!]^\eta_\mu(\mathcal{W})) \Rrightarrow (\lambda\mu.\ \lambda\mathcal{W}.\ [\![Q]\!]^\eta_\mu(\mathcal{W}))
\end{aligned}
$$

3.3 Non-promising safety implies promising safety

Lemma 20.

$$
\begin{aligned}
&\langle\langle \sigma_1, V_1\rangle, \mathit{gconf}_1\rangle \xrightarrow{\text{NP}}{}^*_i \langle\langle \sigma_2, V_2\rangle, \mathit{gconf}_2\rangle \xrightarrow{\text{NP}}{}^*_i \langle\langle \sigma_3, V_3\rangle, \mathit{gconf}_3\rangle \wedge \mathit{gconf}_1.P \subseteq \mathit{gconf}_1.M \\
&\quad \Rightarrow \text{written}(\mathit{gconf}_1, \mathit{gconf}_3) \subseteq \text{written}(\mathit{gconf}_1, \mathit{gconf}_2) \cup \text{written}(\mathit{gconf}_2, \mathit{gconf}_3)
\end{aligned}
$$

Lemma 21.

$$
\begin{aligned}
&\mathit{gconf} \in \lfloor r_F[i \mapsto r \bullet f], u, \mathcal{W} \rfloor_{\{i\}} \wedge (\mathit{gconf}, V, r) \in \text{npsafe}_{(n+1,k)}(\sigma, B)(\mathcal{W}) \wedge \\
&m \in \text{written}(\mathit{gconf}, \mathit{gconf}') \wedge \langle\langle \sigma, V\rangle, \mathit{gconf}\rangle \xrightarrow{\text{NP}}{}^k_i \langle\langle \sigma', V'\rangle, \mathit{gconf}'\rangle \\
&\quad \Rightarrow \text{writeAllowed}(r \bullet \text{canAcq}(r, u), m)
\end{aligned}
$$

Proof. By induction on k.

- Case $k = 0$: by assumption $\mathit{gconf}'.M \neq \mathit{gconf}.M$ or $\mathit{gconf}'.P \neq \mathit{gconf}.P$ which contradicts $\mathit{gconf} = \mathit{gconf}'$.

- Case $k > 0$: then there exists σ'', V'', and $gconf''$ such that

$$\langle\langle \sigma, V\rangle, gconf\rangle \xrightarrow{\text{NP}}_i \langle\langle \sigma'', V''\rangle, gconf''\rangle \xrightarrow{\text{NP}}_i{}^{k-1} \langle\langle \sigma', V'\rangle, gconf'\rangle$$

From the npsafe assumption there exists r'', u'', and $\mathcal{W}'' \geq \mathcal{W}$ such that

$$gconf'' \in \lfloor r_F[i \mapsto r'' \bullet f], u'', \mathcal{W}''\rfloor_{\{i\}} \qquad (gconf'', V'', r'') \in \text{npsafe}_{(n+1,k-1)}(\sigma'', B)(\mathcal{W}'')$$

$r'' \bullet \text{canAcq}(r'', u'') \leq_{\text{o}} r \bullet \text{canAcq}(r, u)$, $\text{writesAllowed}(r, \text{written}(gconf, gconf''))$.

  - Case $m \in \text{written}(gconf, gconf'')$: Then it it follows from the

$$\text{writesAllowed}(r, \text{written}(gconf, gconf''))$$

    assumption that $\text{writeAllowed}(r, m)$ and thus, by upwards-closure that $\text{writeAllowed}(r \bullet \text{canAcq}(r, u), m)$.
  - Case $m \notin \text{written}(gconf, gconf'')$: Then by Lemma 20, $m \in \text{written}(gconf'', gconf')$. By Lemma 16 it suffices to prove that

$$\text{writeAllowed}(r'' \bullet \text{canAcq}(r'', u''), m)$$

    which follows from the induction hypothesis.

□

Lemma 22.

$$pln(m) \Rightarrow \forall x \in PLoc.\ last(M(x) \setminus P) = last((M \hookleftarrow m)(x) \setminus (P \hookleftarrow m))$$

where $\hookleftarrow \in \{\overset{\text{A}}{\hookleftarrow}, \overset{\text{S}}{\hookleftarrow}, \overset{\text{U}}{\hookleftarrow}\}$.

Proof.

- Case $\overset{\text{A}}{\hookleftarrow}$: then $M' = M \hookleftarrow m = M \cup \{m\}$, $P' = P \hookleftarrow m = P \cup \{m\}$, $\{m\}\#M$ and $\{m\}\#P$. Hence, $last(M'|_x \setminus P') = last(M|_x \setminus M)$.
- Case $\overset{\text{S}}{\hookleftarrow}$: then there exists m', m'' such that $M' = M \hookleftarrow m = (M \setminus \{m'\}) \cup \{m, m''\}$, $P' = P \hookleftarrow m = (P \setminus \{m'\}) \cup \{m, m''\}$, where $m' \in M \cap P$ and $m.\texttt{loc} = m'.\texttt{loc} = m''.\texttt{loc}$. Hence, $last(M'|_x \setminus P') = last(M|_x \setminus M)$.
- Case $\overset{\text{U}}{\hookleftarrow}$: then there exists an m' such that $M' = M \hookleftarrow m = (M \setminus \{m'\}) \cup \{m\}$, $P' = P \hookleftarrow m = (P \setminus \{m'\}) \cup \{m\}$, where $m' \in M \cap P$ and $m.\texttt{loc} = m'.\texttt{loc}$. Hence, $last(M'|_x \setminus P') = last(M|_x \setminus M)$.

□

Lemma 23.

$$(M, P) \in \lfloor r_F, u, \mathcal{W}\rfloor_\emptyset \wedge pln(m) \Rightarrow (M \hookleftarrow m, P \hookleftarrow m) \in \lfloor r_F, u, \mathcal{W}\rfloor_\emptyset$$

where $\hookleftarrow \in \{\overset{\text{A}}{\hookleftarrow}, \overset{\text{S}}{\hookleftarrow}, \overset{\text{U}}{\hookleftarrow}\}$.

Proof. Follows from 22. □

Lemma 24.

$$\begin{aligned}&(M, P) \in \lfloor r_F, u, \mathcal{W}\rfloor_\emptyset \wedge m.\texttt{mod} = \texttt{rlx} \wedge \text{validPromise}(r_F, u, m) \wedge \\ &(gconf_\perp, V_\perp, \varepsilon) \in [\![\mathcal{W}.\texttt{rel}(m.\texttt{loc})(m.\texttt{val})]\!]_{[]}(|\mathcal{W}|) \\ &\Rightarrow (M', P') \in \lfloor r_F, u, \mathcal{W}\rfloor_\emptyset\end{aligned}$$

where $\hookleftarrow \in \{\overset{\text{A}}{\hookleftarrow}, \overset{\text{S}}{\hookleftarrow}, \overset{\text{U}}{\hookleftarrow}\}$, $M' = M \hookleftarrow m$ and $P' = P \hookleftarrow m$.

Proof. By case-analysis of $\hookleftarrow$.

- Case $\hookleftarrow^{\text{A}}$: then $M' = M \hookleftarrow m = M \cup \{m\}$, $P' = P \hookleftarrow m = P \cup \{m\}$, $\{m\}\#M$ and $\{m\}\#P$. From the validPromise assumption it follows that validPromises$_\emptyset(r_F, u, P')$. It follows by upwards-closure in the observations component that all existing relaxed messages satisfy the required predicate. It thus remains to prove that newly promised message does as well and that the corresponding predicate is pure, which follows directly from the assumptions.

- Case $\hookleftarrow^{\text{S}}$: then there exists m', m'' such that $M' = M \hookleftarrow m = (M \setminus \{m'\}) \cup \{m, m''\}$, $P' = P \hookleftarrow m = (P \setminus \{m'\}) \cup \{m, m''\}$, where $m' \in M \cap P$, $m.\texttt{loc} = m'.\texttt{loc} = m''.\texttt{loc}$, $m'.\texttt{val} = m''.\texttt{val}$, $m'.\texttt{view} = m''.\texttt{view}$, $m.\texttt{tid} = m'.\texttt{tid} = m''.\texttt{tid}$, and $m.\texttt{mod} = m'.\texttt{mod} = m''.\texttt{mod} = \texttt{rlx}$.

  It follows from the erasure assumption that

  $$(gconf_\bot, V_\bot, \varepsilon) \in [\![\mathcal{W}.\texttt{rel}(m'.\texttt{loc})(m'.\texttt{val})]\!](|\mathcal{W}|)$$

  from which it follows that

  $$(gconf_\bot, V_\bot, \varepsilon) \in [\![\mathcal{W}.\texttt{rel}(m''.\texttt{loc})(m''.\texttt{val})]\!](|\mathcal{W}|)$$

  It follows that all the assertions associated with relaxed messages hold.

  It thus remains to prove that all outstanding promises are valid. Validity for m follows from the lemma assumptions and validity of m'' follows from validity of m' which follows from the erasure assumption.

- Case $\hookleftarrow^{\text{U}}$: then there exists m' such that $M' = M \hookleftarrow m = (M \setminus \{m'\}) \cup \{m\}$, $P' = P \hookleftarrow m = (P \setminus \{m'\}) \cup \{m\}$, where $m' \in M \cap P$, $m.\texttt{loc} = m'.\texttt{loc}$, $m.\texttt{val} = m'.\texttt{val}$, $m.\texttt{view} = m'.\texttt{view}$ and $m.\texttt{tid} = m'.\texttt{tid}$.

  It follows directly from the lemma assumptions that the m promise is valid and that the assertion associated with m holds.

□

Lemma 25 (Non-promising safety implies safety)**.**

$$\forall n, \sigma, B, \mathcal{W}.\ \text{npsafe}_{(n+1,0)}(\sigma, B)(\mathcal{W}) \subseteq \text{safe}_n(\sigma, B)(\mathcal{W})$$

Proof. By induction on n. The $n = 0$ case holds by definition. Suppose that the claim holds for n, and let σ, r such that $(gconf, V, r) \in \text{npsafe}_{(n+2,0)}(\sigma, B)(\mathcal{W})$. We show that $(gconf, V, r) \in \text{safe}_{n+1}(\sigma, B)(\mathcal{W})$ holds. The first conjunct holds since $(gconf, V, r) \in \text{npsafe}_{n+1,1}(\sigma, B)(\mathcal{W})$.

For the second conjunct, assume $gconf' \in \lfloor r_F[i \mapsto r], u, \mathcal{W} \rfloor_\emptyset$, $gconf \leq gconf'$ and

$$\langle\langle \sigma, V\rangle, gconf'\rangle \overset{\beta}{\Longrightarrow}_i \langle\langle \sigma', V'\rangle, gconf''\rangle$$

- Case $\beta =$ promise: then there exists an m such that $\sigma' = \sigma$, $V' = V$, $\langle\langle \sigma', V'\rangle, gconf''\rangle$ is consistent w.r.t. i and

  $$gconf''.M = gconf'.M \hookleftarrow m \qquad gconf''.P = gconf'.P \hookleftarrow m \qquad m.\texttt{tid} = i$$

  $\hookleftarrow \in \{\hookleftarrow^{\text{A}}, \hookleftarrow^{\text{S}}, \hookleftarrow^{\text{U}}\}$.

  Hence, by the definition of consistency there exists σ'', V'', and $gconf'''$ such that $gconf'''.P(i) = \emptyset$ and

  $$\langle\langle \sigma', V'\rangle, gconf''\rangle \xrightarrow{\text{NP}}{}^*_i \langle\langle \sigma'', V''\rangle, gconf'''\rangle$$

  It follows by Corollary 1 that there exists a k such that

  $$\langle\langle \sigma, V\rangle, gconf'\rangle \xrightarrow{\text{NP}}{}^k_i \langle\langle \sigma'', V''\rangle, gconf'''\rangle$$

Since $m \in gconf''.M$, $gconf'''.P(i) = \emptyset$ and $m.\texttt{tid} = i$ it follows that $m \in \text{written}(gconf', gconf''')$.

By downwards-closure in the step-index and upwards-closure in $gconf$ it follows that

$$(gconf', V, r) \in \text{npsafe}_{(n+1,k+1)}(\sigma, B)(\mathcal{W})$$

and by promise weakening it follows that $gconf' \in \lfloor r_F[i \mapsto r], u, \mathcal{W} \rfloor_{\{i\}}$. Hence, by Lemma 21, $\text{writeAllowed}(r \bullet \text{canAcq}(r, u), m)$.

Pick $u' = u$, $\mathcal{W}' = \mathcal{W}$ and $r' = r$. It remains to prove that

$$gconf'' \in \lfloor r_F[i \mapsto r], u, \mathcal{W} \rfloor_{\emptyset} \qquad\qquad (gconf'', V, r) \in \text{safe}_n(\sigma, B)(\mathcal{W})$$

The second proof obligation follows from the induction hypothesis.

- Case $m.\texttt{mod} = \texttt{rlx}$: we proceed using Lemma 24, which requires us to prove that the promise is allowed: $\text{validPromise}(r_F[i \mapsto r], u, m)$ and that the assertion associated with the write holds: $(gconf_\perp, V_\perp, \varepsilon) \in [\![\mathcal{W}.\texttt{rel}(m.\texttt{loc})(m.\texttt{val})]\!]^{[]}_{[]}(|\mathcal{W}|)$. We have already shown that the promise is allowed.

  Applying the npsafe assumption k-times it follows that there exists r', u' $\mathcal{W}' \geq \mathcal{W}$ such that $gconf'' \in \lfloor r_F[i \mapsto r'], u', \mathcal{W}' \rfloor_{\{i\}}$. Thus,

  $$(gconf_\perp, V_\perp, \varepsilon) \in [\![\mathcal{W}'.\texttt{rel}(m.\texttt{loc})(m.\texttt{val})]\!]^{[]}_{[]}(|\mathcal{W}'|)$$

  By world ordering it follows that $\mathcal{W}.\texttt{rel}(\texttt{loc}) = \mathcal{W}'.\texttt{rel}(\texttt{loc})$ and $|\mathcal{W}| = |\mathcal{W}'|$ and thus

  $$(gconf_\perp, V_\perp, \varepsilon) \in [\![\mathcal{W}.\texttt{rel}(m.\texttt{loc})(m.\texttt{val})]\!]^{[]}_{[]}(|\mathcal{W}|)$$

  as required.
- Case $m.\texttt{mod} = \texttt{pln}$: by Lemma 23 and the fact that $\text{writeAllowed}(r \bullet \text{canAcq}(r, u), m)$.
- Case $m.\texttt{mod} = \texttt{rel}$: contradiction, as release write promises cannot be fulfilled.

- Case $\beta = \text{NP}$: then there exists a $gconf'''$ such that $\langle \langle \sigma', V' \rangle, gconf''' \rangle$ is consistent w.r.t. i, such that $gconf''.M = gconf'''.M$, $gconf''.P = gconf'''.P$, and

  $$\langle \langle \sigma, V \rangle, gconf' \rangle \xrightarrow{\text{NP}}{}_i^{+} \langle \langle \sigma', V' \rangle, gconf''' \rangle$$

  By the definition of consistency it follows that there exists σ'', V'', $gconf''''$, k_1, and k_2 such that $gconf''''.P(i) = \emptyset$ and

  $$\langle \langle \sigma, V \rangle, gconf' \rangle \xrightarrow{\text{NP}}{}_i^{k_1+1} \langle \langle \sigma', V' \rangle, gconf''' \rangle \xrightarrow{\text{NP}}{}_i^{k_2} \langle \langle \sigma'', V'' \rangle, gconf'''' \rangle$$

  By downwards-closure of npsafe it follows that

  $$(gconf', V, r) \in \text{npsafe}_{(n+1,k_1+k_2+2)}(\sigma, B)(\mathcal{W})$$

  It follows by $k_1 + 1$ applications of npsafe that there exists r', u' and $\mathcal{W}' \geq \mathcal{W}$ such that

  $$gconf''' \in \lfloor r_F[i \mapsto r'], u', \mathcal{W}' \rfloor_{\{i\}} \qquad (gconf''', V', r') \in \text{npsafe}_{(n+1,k_2+1)}(\sigma', B)(\mathcal{W}')$$

  By downwards-closure of npsafe and the induction hypothesis, it thus follows that $(gconf''', V', r') \in \text{safe}_n(\sigma', B)$.

  It thus remains to show that $promisesAllowed_\emptyset(r_F[i \mapsto r'], u', gconf'''.P)$. Let $m \in gconf'''.P$.

– Case $m.\texttt{tid} = i$: then we have to prove that

$$\text{validPromise}(r', u', m)$$

This follows from Lemma 21, since $m \in \text{written}(gconf''', gconf'''')$.

– Case $m.\texttt{tid} \neq i$: then the conclusion follows from

$$promisesAllowed_{\{i\}}(r_F[i \mapsto r'], u', gconf'''.P).$$

□

3.4 Adequacy of promising safety

Lemma 26.

$$\begin{array}{l}\forall I \in \mathcal{P}_{fin}(TId).\ \forall r, V, B.\\ \quad gconf \in \lfloor r_F[i \in I \mapsto r_i], u, \mathcal{W} \rfloor_\emptyset \wedge\\ \quad \forall i \in I.\ (gconf, V_i, r_i) \in vs(B_i)(\mathcal{W})\\ \qquad \Rightarrow \exists r', u', \mathcal{W}' \geq \mathcal{W}.\\ \qquad\quad gconf \in \lfloor r_F[i \in I \mapsto r'_i], u', \mathcal{W}' \rfloor_\emptyset \wedge\\ \qquad\quad \forall i \in I.\ (gconf, V_i, r'_i) \in B_i(\mathcal{W}')\end{array}$$

Proof. By induction on $|I|$.

- Case $|I| = 0$: follows trivially by picking $r' = r$, $u' = u$ and $\mathcal{W}' = \mathcal{W}$.
- Case $|I| > 0$: pick a $j \in I$. From the assumption that $(gconf, V_j, r_j) \in vs(B_j)(\mathcal{W})$ it follows that there exists r'_j, u' and $\mathcal{W}' \geq \mathcal{W}$ such that

$$gconf \in \lfloor (r_F[j \mapsto r'_j])[i \in I \setminus \{j\} \mapsto r_i], u', \mathcal{W}' \rfloor_\emptyset \qquad\qquad (gconf, V_j, r'_j) \in B_j(\mathcal{W}')$$

Furthermore, by upwards-closure in the world component we have that

$$\forall i \in I \setminus \{j\}.\ (gconf, V_i, r_i) \in vs(B_i)(\mathcal{W}')$$

From the induction hypothesis it thus follows that there exists r'', u'' and $\mathcal{W}'' \geq \mathcal{W}$ such that

$$gconf \in \lfloor (r_F[j \mapsto r'_j])[i \in I \setminus \{j\} \mapsto r''_i], u'', \mathcal{W}'' \rfloor_\emptyset$$

and

$$\forall i \in I \setminus \{j\}.\ (gconf, V_i, r''_i) \in B_i(\mathcal{W}'')$$

The conclusion now follows by picking $u' = u''$, $\mathcal{W}' = \mathcal{W}''$, and

$$r' = [j \mapsto r'_j, i \in I \setminus \{j\} \mapsto r''_i]$$

since $(gconf, V_k, r'_j) \in B_j(\mathcal{W}'')$ by upwards-closure.

□

Lemma 27 (Adequacy)**.** If

1. $(gconf, \Sigma(i).V, r(i)) \in \text{safe}_{n+1}(\Sigma(i).\sigma, B_i)(\mathcal{W})$ for all $i \in dom(\Sigma)$,
2. $gconf \in \lfloor r, u, \mathcal{W} \rfloor_\emptyset$,

3. $(\Sigma, gconf) \Longrightarrow^n (\Sigma', gconf')$,
4. $\forall i \in dom(\Sigma').\ \Sigma'(i).s = \textsf{skip}$

then there exists r', u' and $\mathcal{W}' \geq \mathcal{W}$ such that

1. $gconf' \in \lfloor r', u', \mathcal{W}' \rfloor_\emptyset$,
2. and $(gconf', \Sigma'.V(i), r'(i)) \in [\![B_i]\!]_{\Sigma'(i).\mu}(\mathcal{W}')$ for all $i \in dom(\Sigma')$.

Proof. By induction on n.

- Case $n = 0$: then $(\Sigma, gconf) = (\Sigma', gconf')$ and we can simply pick $r' = r$, $u' = u$ and $\mathcal{W}' = \mathcal{W}$. Since $\Sigma(i).s = \textsf{skip}$ for every $i \in dom(\Sigma)$ it follows from the assumed safety that $(gconf', \Sigma(i).V, r(i)) \in vs(B_i)(\mathcal{W})$ for all $i \in dom(\Sigma)$. The conclusion now follows from Lemma 26.
- Case $n > 0$: then there exists Σ'' and $gconf''$ such that

  $$(\Sigma, gconf) \Longrightarrow_i (\Sigma'', gconf'') \Longrightarrow^{n-1} (\Sigma', gconf')$$

  From the safety assumption it follows that there exists r', u' and $\mathcal{W}' \geq \mathcal{W}$ such that $gconf'' \in \lfloor r[i \mapsto r'], u', \mathcal{W}' \rfloor_\emptyset$ and

  $$(gconf'', \Sigma''(i).V, r') \in \text{safe}_{n-1}(\Sigma''(i).\sigma, B_i)(\mathcal{W}')$$

  The conclusion follows by the induction hypothesis and upwards-closure of safety and erasure in the world component and upwards-closure of safety in $gconf$.

□

4 Soundness

In this section we prove soundness of the syntactic proof system with respect to the non-promising semantics.

4.1 Structural rules

Lemma 28 (Soundness of sequential composition)**.**

$$\forall n, m.\ P(n, m)$$

where $P(n, m)$ is defined as follows

$$\begin{array}{l}\forall r_1, \mu_1, s_1, s_2, \mathcal{W}_1. \\ \quad (\forall \mu_2, \mathcal{W}_2 \geq \mathcal{W}_1.\ B(\mu_2)(\mathcal{W}_2) \subseteq \mathrm{npsafe}_{(n,m)}((\mu_2, s_2), C)(\mathcal{W}_2)) \\ \quad \Rightarrow \mathrm{npsafe}^i_{(n,m)}((\mu_1, s_1), B)(\mathcal{W}_1) \subseteq \mathrm{npsafe}_{(n,m)}((\mu_1, s_1; s_2), C)(\mathcal{W}_1)\end{array}$$

Proof. By induction on (n, m).

Assume $P(n', m')$ holds for all $(n', m') < (n, m)$.

- Case $n = 0$: then $P(n, m)$ holds vacuously.
- Case $n > 0$ and $m = 0$: then we can assume that

  $$\forall k.\ (\mathit{gconf}, V, r) \in \mathrm{npsafe}_{(n-1,k)}((\mu_1, s_1), B)(\mathcal{W}_1)$$

  and

  $$\forall \mu_2.\ \forall \mathcal{W}_2 \geq \mathcal{W}_1.\ \forall k.\ B(\mu_2)(\mathcal{W}_2) \subseteq \mathrm{npsafe}_{(n-1,k)}((\mu_2, s_2), C)(\mathcal{W}_2)$$

  and must prove $\forall k.\ (\mathit{gconf}, V, r) \in \mathrm{npsafe}_{(n-1,k)}((\mu_1, s_1; s_2), C)$. This follows easily by the induction hypothesis.
- Case $n > 0$ and $m > 0$: Assume

  $$(\mathit{gconf}, V, r) \in \mathrm{npsafe}_{(n,m)}((\mu_1, s_1), B)(\mathcal{W}_1)$$

  and

  $$\forall \mu_2, \mathcal{W}_2 \geq \mathcal{W}_1.\ B(\mu_2)(\mathcal{W}_2) \subseteq \mathrm{npsafe}^i_{(n,m)}((\mu_2, s_2), C)(\mathcal{W}_2).$$

  Since $s_1; s_2 \neq \textsf{skip}$ it suffices to consider the case where $\mathit{gconf} \in \lfloor r_F[i \mapsto r \bullet f], u, \mathcal{W} \rfloor_{\{i\}}$, $\mathcal{W} \geq \mathcal{W}_1$, $\mathrm{wf}_{\mathrm{conf}}(i, \mathit{gconf}, V)$, $\mathrm{wf}_{\mathrm{conf}}(i, \mathit{gconf}', V')$ and

  $$\langle\langle(\mu_1, s_1; s_2), V\rangle, \mathit{gconf}\rangle \xrightarrow{\mathrm{NP}}_i \langle\langle(\mu', s'), V'\rangle, \mathit{gconf}'\rangle)$$

  - Case $s_1 = \textsf{skip}$: then $\mu' = \mu_1$, $s' = s_2$, $V' = V$ and $\mathit{gconf}' = \mathit{gconf}$. Hence, $(\mathit{gconf}, V, r) \in \mathit{vs}(B(\mu_1))(\mathcal{W})$. There thus exists r', u' and $\mathcal{W}' \geq \mathcal{W}$ such that

    $$(\mathit{gconf}, V, r') \in B(\mu_1)(\mathcal{W}') \quad \mathit{gconf} \in \lfloor r_F[i \mapsto r' \bullet f], u', \mathcal{W}' \rfloor_{\{i\}} \quad r' \leq_{\mathrm{o}} r \quad r' \bullet \mathrm{canAcq}(r', u') \leq_{\mathrm{o}} r \bullet \mathrm{canAcq}(r, u)$$

    and thus $(\mathit{gconf}, V, r') \in \mathrm{npsafe}_{(n,m)}((\mu', s'), C)(\mathcal{W}')$. The conclusion now follows by picking r', u' and $\mathcal{W}'$, since $\mathrm{written}(\mathit{gconf}, \mathit{gconf}') = \emptyset$.

– Case $s_1 \neq \textsf{skip}$: then there exists an s_1' such that

$$\langle\langle(\mu_1, s_1), V\rangle, gconf\rangle \xrightarrow{\text{NP}}_i \langle\langle(\mu', s_1'), V'\rangle, gconf'\rangle$$

and $s' = s_1'; s_2$. Hence, by the assumed safety for s_1 there exists an r', u' and $\mathcal{W}' \geq \mathcal{W}$ such that

$$gconf' \in \lfloor r_F[i \mapsto r' \bullet f], u', \mathcal{W}\rfloor_{\{i\}} \qquad (gconf', V', r') \in \text{npsafe}_{(n,m-1)}((\mu', s_1'), B)(\mathcal{W}')$$

and

$$r' \bullet \text{canAcq}(r', u') \leq_{\text{o}} r \bullet \text{canAcq}(r, u) \qquad \text{writesAllowed}(r, \text{written}(gconf, gconf'))$$

By downwards-closure and the induction hypothesis, it follows that

$$(gconf', V', r') \in \text{npsafe}_{(n,m-1)}((\mu', s_1'; s_2), C)(\mathcal{W}').$$

Now the conclusion follows easily by picking r', u' and $\mathcal{W}'$.

□

Lemma 29 (Soundness of while-rule)**.**

$$[\![\vdash \{P \wedge e = \top\}\ s\ \{P\}]\!]^{\text{np}} \Rightarrow [\![\vdash \{P\}\ \textsf{while}\ e\ \textsf{do}\ s\ \{P \wedge e = \bot\}]\!]^{\text{np}}$$

Proof. Assume $[\![\vdash \{P \wedge e = \top\}\ s\ \{P\}]\!]^{\text{np}}$. We will prove that $\forall n, m.\ P(n, m)$ holds by induction on (n, m), where $P(n, m)$ is defined as follows.

$$P(n,m) \stackrel{def}{=} \forall \mu, \eta, \mathcal{W}.\ [\![P]\!]^{\eta}_{\mu}(\mathcal{W}) \subseteq \text{npsafe}_{(n,m)}((\mu, \textsf{while}\ e\ \textsf{do}\ s), \lambda\mu.\ \lambda\mathcal{W}.\ [\![P \wedge e = \bot]\!]^{\eta}_{\mu}(\mathcal{W}))(\mathcal{W})$$

Assume $P(n', m')$ holds for all $(n', m') < (n, m)$. To show that $P(n, m)$ holds, assume $(gconf, V, r) \in [\![P]\!]^{\eta}_{\mu}(\mathcal{W})$.

- Case $n = 0$: trivial.
- Case $n > 0$ and $m = 0$: follows directly from the induction hypothesis.
- Case $n > 0$ and $m > 0$: assume $gconf \in \lfloor r_F[i \mapsto r \bullet f], u, \mathcal{W}\rfloor_{\{i\}}$ and

$$\langle\langle(\mu, \textsf{while}\ e\ \textsf{do}\ s), V\rangle, gconf\rangle \xrightarrow{\text{NP}}_i \langle\langle(\mu', s'), V'\rangle, gconf'\rangle$$

Then $\mu = \mu'$, $V' = V$, and $gconf' = gconf$.

– Case $[\![e]\!]_{\mu} = \top$: then $s' = s; \textsf{while}\ e\ \textsf{do}\ s$. By assumption, $P(n, m-1)$ and thus, by Lemma 28,

$$\begin{aligned}&\text{npsafe}_{(n,m-1)}((\mu, s), \lambda\mu.\ \lambda\mathcal{W}.\ [\![P]\!]^{\eta}_{\mu}(\mathcal{W}))(\mathcal{W})\\ &\subseteq \text{npsafe}_{(n,m-1)}((\mu, s; \textsf{while}\ e\ \textsf{do}\ s), \lambda\mu.\ \lambda\mathcal{W}.\ [\![P \wedge e = \bot]\!]^{\eta}_{\mu}(\mathcal{W}))(\mathcal{W})\end{aligned}$$

Furthermore, from the assumed triple for the loop body it follows that

$$(gconf, V, r) \in \text{npsafe}_{(n,m-1)}((\mu, s), \lambda\mu.\ \lambda\mathcal{W}.\ [\![P]\!]^{\eta}_{\mu}(\mathcal{W}))(\mathcal{W}).$$

Now the conclusion follows easily by picking r, u and $\mathcal{W}$.

– Case $[\![e]\!]_{\mu} = \bot$: then $s' = \textsf{skip}$ and the conclusion follows easily as $(gconf, V, r) \in [\![P \wedge e = \bot]\!]^{\eta}_{\mu}(\mathcal{W}) \subseteq vs(\lambda\mathcal{W}.\ [\![P \wedge e = \bot]\!]^{\eta}_{\mu}(\mathcal{W}))(\mathcal{W})$.

□

Lemma 30 (Soundness of precondition strengthening)**.**

$$\models A \Rrightarrow A' \wedge \models_{np} \{A'\}\ s\ \{B\} \Rightarrow \models_{np} \{A\}\ s\ \{B\}$$

Proof. Assume

$$(gconf, V, r) \in A(\mu)(\mathcal{W}) \qquad gconf \in \lfloor r_F[i \mapsto r\bullet], u, \{i\} \rfloor_{\mathcal{W}} \qquad \langle\langle(\mu, s), V\rangle, gconf\rangle \xrightarrow{\text{NP}}_i \langle\langle\sigma', V'\rangle, gconf'\rangle$$

From the assumed view-shift it follows that there exists r', u' and $\mathcal{W}' \geq \mathcal{W}$ such that $r' \leq_{\text{o}} r$, $(gconf, V, r') \in [\![A']\!]_\mu(\mathcal{W}')$, and

$$gconf \in \lfloor r_F[i \mapsto r' \bullet f], u', \mathcal{W}' \rfloor_{\{i\}} \qquad\qquad r' \bullet \text{canAcq}(r', u') \leq_{\text{o}} r \bullet \text{canAcq}(r, u)$$

Hence, from the assumed triple it follows that there exists r'', u'' and $\mathcal{W}'' \geq \mathcal{W}'$ such that $(gconf', V', r'') \in \text{npsafe}_{(n,m)}(\sigma', B)(\mathcal{W}'')$,

$$gconf' \in \lfloor r_F[i \mapsto r'' \bullet f], u'', \mathcal{W}'' \rfloor_{\{i\}} \qquad\qquad r'' \bullet \text{canAcq}(r'', u'') \leq_{\text{o}} r' \bullet \text{canAcq}(r', u')$$

and $\text{writesAllowed}(r', \text{written}(gconf, gconf'))$. Since $\leq_{\text{o}}$ is transitive it follows that

$$r'' \bullet \text{canAcq}(r'', u'') \leq_{\text{o}} r \bullet \text{canAcq}(r, u).$$

It thus remains to prove that $\text{writesAllowed}(r, \text{written}(gconf, gconf'))$. This follows from $r' \leq_{\text{o}} r$ by Lemma 16 and the fact that $\text{writesAllowed}(r', \text{written}(gconf, gconf'))$. □

Lemma 31.

$$\forall \mathcal{W}.\ vs(vs(B))(\mathcal{W}) \subseteq vs(B)(\mathcal{W})$$

Proof. Follows from transitivity of $\leq_{\text{o}}$ and the order on worlds. □

Lemma 32.

$$\forall \mathcal{W}.\ \text{npsafe}_{(n,m)}(\sigma, \lambda\mu.\ vs(B(\mu)))(\mathcal{W}) \subseteq \text{npsafe}_{(n,m)}(\sigma, B)(\mathcal{W})$$

Proof. By well-founded induction, using the induction hypothesis for reduction steps and Lemma 31 for the skip case. □

Lemma 33.

$$\begin{aligned}&(\forall \mu, \mathcal{W}.\ p(\mu)(\mathcal{W}) \subseteq q(\mu)(\mathcal{W})) \Rightarrow \\ &\quad \forall n, m, \sigma, \mathcal{W}.\ \text{npsafe}_{(n,m)}(\sigma, p)(\mathcal{W}) \subseteq \text{npsafe}_{(n,m)}(\sigma, q)(\mathcal{W})\end{aligned}$$

Lemma 34 (Soundness of postcondition weakening)**.**

$$\begin{aligned}&\forall B, B'.\ \models B' \Rrightarrow B \Rightarrow \\ &\quad \forall n, m, \sigma, \mathcal{W}.\ \text{npsafe}_{(n,m)}(\sigma, B')(\mathcal{W}) \subseteq \text{npsafe}_{(n,m)}(\sigma, B)(\mathcal{W})\end{aligned}$$

Proof. By Lemmas 33 and 32 and the definition of viewshifts it follows that

$$\begin{aligned}\text{npsafe}_{(n,m)}(\sigma, B')(\mathcal{W}) &\subseteq \text{npsafe}_{(n,m)}(\sigma, \lambda\mu.\ vs(B(\mu)))(\mathcal{W}) \\ &\subseteq \text{npsafe}_{(n,m)}(\sigma, B)(\mathcal{W})\end{aligned}$$

as required. □

Lemma 35.

$$(\forall \mathcal{W}.\ p(\mathcal{W}) \subseteq q(\mathcal{W})) \Rightarrow \forall \mathcal{W}.\ vs(p)(\mathcal{W}) \subseteq vs(q)(\mathcal{W})$$

4.2 View-shift rules

Lemma 36 (Splitting of acquire permissions)**.**

$$[\![\mathsf{Acq}(x, \lambda v.\ \phi_1(v) * \phi_2(v)) \Rrightarrow \mathsf{Acq}(x, \phi_1) * \mathsf{Acq}(x, \phi_2)]\!]$$

Proof. Assume $(gconf_1, V_1, r_1) \leq (gconf, V, r)$ and

$$(gconf_1, V_1, r_1) \in [\![\mathsf{Acq}(x, \lambda v.\ \phi_1(v) * \phi_2(v))]\!]^\eta_\mu(\mathcal{W}) \qquad gconf \in \lfloor r_F[i \mapsto r \bullet f], u, \mathcal{W} \rfloor_T$$

Then there exists an $\iota \in dom(\mathcal{W}.\mathtt{acq}) \cap r.\mathtt{acq}([\![x]\!]^\eta_\mu)$ such that

$$\mathcal{W}.\mathtt{acq}(\iota) = \lambda v.\ \phi_1(v) * \phi_2(v).$$

Hence, for every $m \in gconf.M(\mathtt{rel}, [\![x]\!]^\eta_\mu)$ we have that

$$(gconf, m.\mathtt{view}, u(m)(\iota)) \in [\![\mathcal{W}.\mathtt{acq}(\iota)(m.\mathtt{val})]\!]^{[]}_{[]}(\mathcal{W}) = [\![\phi_1(m.\mathtt{val}) * \phi_2(m.\mathtt{val})]\!]^{[]}_{[]}(\mathcal{W})$$

Hence, there exists $(r^1_m)_{m \in gconf.M(\mathtt{rel}, x)}$ and $(r^2_m)_{m \in gconf.M(\mathtt{rel}, [\![x]\!]^\eta_\mu)}$ such that

$$(m.\mathtt{view}, r^1_m) \in [\![\phi_1(m.\mathtt{val})]\!]^{[]}_{[]}(\mathcal{W}) \qquad (m.\mathtt{view}, r^2_m) \in [\![\phi_2(m.\mathtt{val})]\!]^{[]}_{[]}(\mathcal{W})$$

and $u(m)(\iota) = r^1_m \bullet r^2_m$ for all $m \in npln(gconf.M(x))$.

Pick fresh $\iota_1, \iota_2 \notin dom(\mathcal{W}.\mathtt{acq})$ and let

$$\begin{aligned}
\mathcal{W}' &= \mathcal{W}[\iota_1 \mapsto \phi_1, \iota_2 \mapsto \phi_2] \\
u' &= u[m \in npln(gconf.M(x)) \mapsto u(m)[\iota \mapsto \bot, \iota_1 \mapsto r^1_m, \iota_2 \mapsto r^2_m]] \\
r' &= r[\mathtt{acq}(x) \mapsto (r.\mathtt{acq}(x) \setminus \{\iota\}) \cup \{\iota_1, \iota_2\}]
\end{aligned}$$

It remains to show that

$$(gconf, V, r') \in [\![\mathsf{Acq}(x, \phi_1) * \mathsf{Acq}(x, \phi_2)]\!]^\eta_\mu(\mathcal{W}') \qquad gconf \in \lfloor r_F[i \mapsto r' \bullet f], u', \mathcal{W}' \rfloor_T$$

and $r' \bullet \mathrm{canAcq}(r', u') \leq_o r \bullet \mathrm{canAcq}(r, u)$, $r' \leq_o r$.

The first proof obligation is easily seen to hold. The second proof obligation reduces to proving that $promisesAllowed_T(r_F[i \mapsto r' \bullet f], u', gconf.P)$. Since $\iota \notin r_F(j).\mathtt{acq}$ for all $j \neq i$ this reduces to proving that

$$\forall m \in gconf.P.\ m.\mathtt{tid} = i \wedge i \notin T \Rightarrow \mathrm{writeAllowed}(r' \bullet \mathrm{canAcq}(r', u'), m)$$

This follows from the $promisesAllowed_T(r_F[i \mapsto r \bullet f], u, gconf.P)$ assumption since $\mathrm{canAcq}(r', u') = \mathrm{canAcq}(r, u)$, by Lemma 14.

From $\mathrm{canAcq}(r', u') = \mathrm{canAcq}(r, u)$ it also follows that $r' \bullet \mathrm{canAcq}(r', u') \leq_o r \bullet \mathrm{canAcq}(r, u)$.

It thus remains to prove that $r' \leq_o r$. This follows easily as $r.\mathtt{pln} = r'.\mathtt{pln}$ and $r.\mathtt{wr} = r'.\mathtt{wr}$. □

Lemma 37.

$$[\![\vdash \mathsf{W}^1(x, X) * \mathsf{O}(x, a, t) \Rrightarrow \mathsf{W}^1(x, X) * \mathsf{O}(x, a, t) * (a, t) \in X]\!]$$

Proof. Assume $(gconf_1, V_1, r_1) \in [\![\mathsf{W}^1(x, X) * \mathsf{O}(x, a, t)]\!]^\eta_\mu(\mathcal{W})$, $(gconf, V, r) \geq (gconf_1, V_1, r_1)$, $\mathcal{W}' \geq \mathcal{W}$, $gconf \in \lfloor r_F[i \mapsto r \bullet f], u, \mathcal{W}' \rfloor_f\{i\}$ and $\mathrm{wf}_{\mathrm{conf}}(i, gconf, V)$.

Hence, there exists an $m \in gconf_1.M$ such that $m.\mathtt{val} = [\![a]\!]^\eta_\mu$, $m.\mathtt{time} = [\![t]\!]^\eta_\mu$, $m.\mathtt{loc} = [\![x]\!]^\eta_\mu$, $r.\mathtt{wr}([\![x]\!]^\eta_\mu) = (1, [\![X]\!]^\eta_\mu)$, and $snd(\max([\![X]\!]^\eta_\mu)) \leq V.\mathtt{rlx}([\![x]\!]^\eta_\mu)$. Hence, by erasure, $\{(m.\mathtt{val}, m.\mathtt{time}) \mid m \in seq(gconf.M([\![x]\!]^\eta_\mu) \setminus gconf.P)\} = X$. It thus suffices to prove that $m \notin gconf.P$.

If $m \in gconf.P$ and $m.\mathtt{tid} \neq i$ then it follows by erasure that $\mathrm{validPromise}(r_F, u, m)$. Hence, $snd((r_F(m.\mathtt{tid}) \bullet \mathrm{canAcq}(r_F(m.\mathtt{tid}, u))).\mathtt{wr}([\![x]\!]^\eta_\mu)) > 0$, which is a contradiction since $snd(r.\mathtt{wr}([\![x]\!]^\eta_\mu)) = 1$. If $m \in gconf.P$ and $m.\mathtt{tid} = i$ then it follows by the $\mathrm{wf}_{\mathrm{conf}}$ assumption that $V.\mathtt{rlx}([\![x]\!]^\eta_\mu) < m.\mathtt{time}$, which is a contradiction. □

Lemma 38.

$$\begin{aligned}&\forall \pi_1, \pi_2, X_1, X_2.\ \pi_1 + \pi_2 \leq 1 \land 0 < \pi_1 \leq 1 \land 0 < \pi_2 \leq 1 \Rightarrow \\ &\quad [\![\mathsf{W}_{\pi_1}(x, X_1) * \mathsf{W}_{\pi_2}(x, X_2) \vdash \mathsf{W}_{\pi_1+\pi_2}(x, X_1 \cup X_2)]\!] \land \\ &\quad [\![\mathsf{W}_{\pi_1+\pi_2}(x, X_1 \cup X_2) \vdash \mathsf{W}_{\pi_1}(x, X_1) * \mathsf{W}_{\pi_2}(x, X_2)]\!]\end{aligned}$$

Proof. Follow directly from the monoid definition. □

Lemma 39.

$$[\![\vdash \mathsf{W}^{\pi}(x, X) * (v, t) \in X * (v', t') \in X * v \neq v' \Rrightarrow \mathsf{W}^{\pi}(x, X) * t \neq t']\!]$$

Proof. Assume $(gconf_1, V_1, r_1) \in [\![\mathsf{W}^{\pi}(x, X) * (v, t) \in X * (v', t') \in X * v \neq v']\!]^{\eta}_{\mu}(\mathcal{W})$, $(gconf, V, r) \geq (gconf_1, V_1, r_1)$, $\mathcal{W}' \geq \mathcal{W}$, $gconf \in \lfloor r_F[i \mapsto r \bullet f], u, \mathcal{W}' \rfloor_{\{i\}}$ and $\mathrm{wf}_{\mathrm{conf}}(i, gconf, V)$.

By erasure, $[\![X]\!]^{\eta}_{\mu} \subseteq gconf.M([\![x]\!]^{\eta}_{\mu})$ and the conclusion now follows from the definition of a memory. □

Lemma 40.

$$[\![\vdash \mathsf{W}^{\pi}(x, X) * (\_, t) \in X * (\_, t') \in X \Rrightarrow \mathsf{W}^{\pi}(x, X) * t < t' \lor t = t' \lor t' < t]\!]$$

Proof. Follows from the assumption that the timestamp order is total. □

Lemma 41.

$$[\![\vdash \mathsf{W}^{\pi}(x, X) * (v, t) \in X \Rrightarrow \mathsf{W}^{\pi}(x, X) * \mathsf{O}(x, v, t)]\!]$$

Proof. Assume $(gconf_1, V_1, r_1) \in [\![\mathsf{W}^{\pi}(x, X) * (v, t) \in X]\!]^{\eta}_{\mu}(\mathcal{W})$, $(gconf, V, r) \geq (gconf_1, V_1, r_1)$, $\mathcal{W}' \geq \mathcal{W}$, $gconf \in \lfloor r_F[i \mapsto r \bullet f], u, \mathcal{W}' \rfloor_{\{i\}}$ and $\mathrm{wf}_{\mathrm{conf}}(i, gconf, V)$. By erasure, $[\![X]\!]^{\eta}_{\mu} \subseteq gconf.M([\![x]\!]^{\eta}_{\mu})$. □

4.3 Rules for release/acquire accesses

Lemma 42.

$$\iota \in r_F(t).A(x) \cap dom(\mathcal{W}) \wedge gconf \in \lfloor r_F, u, \mathcal{W} \rfloor_T \wedge \iota' \text{ fresh for } u \text{ and } r_F \\ \Rightarrow gconf \in \lfloor r_F[\iota \mapsto_x \iota'], u[\iota \mapsto_x \iota'], \mathcal{W}[\iota' \mapsto \mathcal{W}(\iota)[a \mapsto \top]] \rfloor_T$$

Proof. Should follow from Lemmas 13 and 11. □

Lemma 43.

$$m \in dom(u) \wedge \iota \in r_F(t).A(x) \cap dom(u(m)) \wedge \mathcal{W}(\iota)(m.\mathtt{val}) = \top \wedge \\ gconf \in \lfloor r_F, u, \mathcal{W} \rfloor_T \wedge \iota' \text{ fresh for } u \text{ and } r_F \\ \Rightarrow gconf \in \lfloor r_F[t \mapsto r_F(t) \bullet u(m)(\iota)], u[m, \iota \mapsto \varepsilon], \mathcal{W} \rfloor_T$$

Lemma 44 (Soundness of acquire read)**.**

$$\left[\!\left[\vdash \{x = e * \mathsf{Acq}(x, \phi) * \mathsf{O}(x, \_, t)\}\ a := [e]_{\mathsf{acq}}\ \{\exists t' \geq t.\ \mathsf{Acq}(x, \phi[a \mapsto \top]) * \phi(a) * \mathsf{O}(x, a, t')\} \right]\!\right]^{\mathrm{np}}$$

where x and t are assumed to be specification variables.

Proof. Let $(gconf, V, r) \in [\![x = e * \mathsf{Acq}(x, \phi) * \mathsf{O}(x, \_, t)]\!]^{\eta}_{\mu}(\mathcal{W})$. Since $a := [e]_{\mathsf{acq}} \neq \textbf{skip}$ it suffices to consider the case where

$$\langle\langle (\mu, a := [e]_{\mathsf{acq}}), V\rangle, gconf\rangle \xrightarrow{\mathrm{NP}}_i \langle\langle \sigma', V'\rangle, gconf'\rangle,$$

and $gconf \in \lfloor r_F[i \mapsto r \bullet f], u, \mathcal{W}' \rfloor_{\{i\}}$, $\mathcal{W}' \geq \mathcal{W}$.

By the Thread: Read rule it follows that there exists an $m = \langle \eta(x) :^{o}_{j} v, V_m @ (\_, t_m]\rangle \in M$ such that $V.\mathtt{rlx}(\eta(x)) \leq t_m$, $gconf'.P = gconf.P$, $gconf'.M = gconf.M$, $V' = V \sqcup [pln : \{\eta(x)@t_m\}, rlx : \{\eta(x)@t_m\}] \sqcup V_m$ and $\sigma' = (\mu[a \mapsto v], \textbf{skip})$.

From the $(gconf, V, r) \in [\![x = e * \mathsf{Acq}(x, \phi) * \mathsf{O}(x, \_, t)]\!]^{\eta}_{\mu}(\mathcal{W})$ assumption it follows that there exists an $\iota \in dom(\mathcal{W}.\mathtt{acq}) \cap r.\mathtt{acq}(\eta(x))$ such that $\mathcal{W}.\mathtt{acq}(\iota) = \phi$ and $\eta(t) \leq V.\mathtt{rlx}(\eta(x))$. Hence, $\eta(t) \leq t_m$.

- Case $o = \mathtt{rlx}$: it follows from the definition of erasure that $(gconf_{\perp}, V_{\perp}, \varepsilon) \in [\![\mathcal{W}.\mathtt{rel}(m.\mathtt{loc})(m.\mathtt{val})]\!]^{[]}_{[]}(|\mathcal{W}'|)$.

  Pick an ι' that is fresh for u and $\mathcal{W}'$. Pick $u' = u[\iota \rightsquigarrow_{\eta(x)} \iota']$, $r' = r[\iota \rightsquigarrow_{\eta(x)} \iota']$ and $\mathcal{W}'' = \mathcal{W}'[\iota' \mapsto \phi[v \mapsto \top]]$. Then it remains to prove that $gconf' \in \lfloor r_F[i \mapsto r' \bullet f], u', |\mathcal{W}''| \rfloor_{\{i\}}$,

  $$(gconf', V', r') \in [\![\exists t' \geq t.\ \mathsf{Rel}(x, \phi[a \mapsto \top]) * \phi(a) * \mathsf{O}(x, a, t')]\!]^{\eta}_{\sigma'.\mu}(|\mathcal{W}''|)$$

  and $r' \bullet \mathrm{canAcq}(r', u') \leq_{\mathrm{o}} r \bullet \mathrm{canAcq}(r, u)$ and $\mathrm{writesAllowed}(r', \mathrm{written}(gconf, gconf'))$.

  The erasure obligation follows by upwards-closure and Lemma 42. The two last obligations follow easily as $r' \bullet \mathrm{canAcq}(r', u') = r' \bullet \mathrm{canAcq}(r, u) \leq_{\mathrm{o}} r \bullet \mathrm{canAcq}(r, u)$ by Lemma 13 and $\mathrm{written}(gconf, gconf') = \emptyset$.

  It thus remains to prove that

  $$(gconf', V', r') \in [\![\exists t' \geq t.\ \mathsf{Rel}(x, \phi[a \mapsto \top]) * \phi(a) * \mathsf{O}(x, a, t')]\!]^{\eta}_{\sigma'.\mu}(\mathcal{W}'').$$

  It follows from the erasure assumption that

  $$\forall v \vdash \mathcal{W}.\mathtt{rel}(\eta(x))(v) \Rightarrow \circledast_{\iota \in t.\mathtt{acq}(\eta(x))} \mathcal{W}.\mathtt{acq}(\iota)(v)$$

where t is the composition of $r_F[i \mapsto r \bullet f]$ and the resources from u. So, in particular, $\iota \in t.\mathtt{acq}(\eta(x))$ since by assumption $\iota \in r.\mathtt{acq}(\eta(x))$ and thus,

$$\vdash \mathcal{W}.\mathtt{rel}(\eta(x))(v) \Rightarrow \phi(v)$$

It follows that $(gconf_\bot, V_\bot, \varepsilon) \in [\![\phi(m.\mathtt{val})]\!]^{[]}_{[]}(|\mathcal{W}'|)$. By upwards-closure and the definition of the interpretation it follows easily that

$$(gconf', V', r') \in [\![\exists t' \geq t.\ \mathsf{Rel}(x, \phi[a \mapsto \top]) * \phi(a) * \mathsf{O}(x, a, t')]\!]^{\eta}_{\sigma'.\mu}(\mathcal{W}'').$$

- Case $o = \mathtt{rel}$: from the definition of erasure it follows that

$$(gconf, V_m, u(m)(\iota)) \in [\![\phi(v)]\!]^{[]}_{[]}(\mathcal{W}')$$

since $\mathcal{W}.\mathtt{acq}(\iota) = \phi$, $m.\mathtt{view} = V_m$ and $m.\mathtt{val} = v$.

Pick an ι' that is fresh for u and $\mathcal{W}'$. Pick $u' = (u[\iota \rightsquigarrow_{\eta(x)} \iota'])[m, \iota' \mapsto \varepsilon]$, $r' = r[\iota \rightsquigarrow_{\eta(x)} \iota'] \bullet u(m)(\iota)$ and $\mathcal{W}'' = \mathcal{W}'[\iota' \mapsto \phi[v \mapsto \top]]$. Then it remains to prove that $gconf' \in \lfloor r_F[i \mapsto r' \bullet f], u', \mathcal{W}'' \rfloor_{\{i\}}$,

$$(gconf', V', r') \in [\![\exists t' \geq t.\ \mathsf{Rel}(x, \phi[a \mapsto \top]) * \phi(a) * \mathsf{O}(x, a, t')]\!]^{\eta}_{\sigma'.\mu}(\mathcal{W}'')$$

and $r' \bullet \mathrm{canAcq}(r', u') \leq_{\mathrm{o}} r \bullet \mathrm{canAcq}(r, u)$ and $\mathrm{writesAllowed}(r, \mathrm{written}(gconf, gconf'))$.

The writesAllowed obligation follows trivially as $\mathrm{written}(gconf, gconf') = \emptyset$ and the

$$(gconf', V', r') \in [\![\exists t' \geq t.\ \mathsf{Rel}(x, \phi[a \mapsto \top]) * \phi(a) * \mathsf{O}(x, a, t')]\!]^{\eta}_{\sigma'.\mu}(\mathcal{W}'')$$

obligation follows by upwards-closure and the definition of the interpretation.

By Lemmas 7 and 11,

$$\begin{aligned} r' \bullet \mathrm{canAcq}(r', u') &= r[\iota \rightsquigarrow_x \iota'] \bullet \mathrm{canAcq}(r[\iota \rightsquigarrow_x \iota'], u[\iota \rightsquigarrow_x \iota']) \\ &= r[\iota \rightsquigarrow_x \iota'] \bullet \mathrm{canAcq}(r, u) \\ &\leq_{\mathrm{o}} r \bullet \mathrm{canAcq}(r, u) \end{aligned}$$

It remains to prove that $gconf'$ is in the erasure: $gconf' \in \lfloor r_F[i \mapsto r' \bullet f], u', \mathcal{W}'' \rfloor_{\{i\}}$. By Lemma 42 it follows that $gconf' \in \lfloor r_F[i \mapsto (r \bullet f)[\iota \rightsquigarrow_x \iota']], u[\iota \rightsquigarrow_x \iota'], \mathcal{W}'' \rfloor_{\{i\}}$ and by Lemma 43, $gconf' \in \lfloor r_F[i \mapsto r' \bullet f], u', \mathcal{W}'' \rfloor_{\{i\}}$ as required.

□

Lemma 45.

$$\begin{aligned} &gconf \in \lfloor r_F[i \mapsto r \bullet f], u, \mathcal{W} \rfloor_{\{i\}} \wedge r.\mathtt{wr}(m.\mathtt{loc}) = (1, X) \wedge \\ &\pi_1(\max(X)) \leq V.\mathtt{rlx}(m.\mathtt{loc}) \leq m.\mathtt{time} \wedge m \in gconf.M \wedge \\ &\mathrm{wf}_{\mathrm{conf}}(i, gconf, V \sqcup [\mathtt{rlx} : \{m.\mathtt{loc}@m.\mathtt{time}\}]) \Rightarrow \\ &\quad m.\mathtt{val} = \pi_1(\max(X)) \end{aligned}$$

Proof. Since $(1, X)$ is only composable with $(0, \emptyset)$ it follows by the definition of erasure that $gconf.M(m.\mathtt{loc}) \setminus gconf.P = X$. If $m \notin gconf.P$ it follows from the assumptions that $m \in X$ and thus $\max(X) = (m.\mathtt{val}, m.\mathtt{time})$. Otherwise, $m \in gconf.P$. If $m.\mathtt{tid} \neq i$ it follows from erasure that $\mathrm{validPromise}(r_F, u, m)$. Hence, $\pi_1(r_F(m.\mathtt{tid}) \bullet \mathrm{canAcq}(r_F(m.\mathtt{tid}, u))).\mathtt{wr}(x) \neq 0$, which is a contradiction since $r.\mathtt{wr}(x) = (1, X)$. If $m.\mathtt{tid} = i$ then it follows by the $\mathrm{wf}_{\mathrm{conf}}$ assumption that $m.\mathtt{time} < m.\mathtt{time}$ which is a contradiction. □

Lemma 46 (Soundness of acquire read for an exclusive writer)**.**

$$\models_{np} \{x = e * \mathsf{Acq}(x, \phi) * \mathsf{W}^1(x, X)\}\ a := [e]_{\mathsf{acq}}\ \{B\}$$

where x and X are specification variables and B is the assertion

$$\mathsf{Acq}(x, \phi[a \mapsto \top]) * \phi(a) * \mathsf{W}^1(x, X) * \mathsf{O}(x, a, \mathit{snd}(\max(X))) * a = \mathit{fst}(\max(X)).$$

Proof. The same as Lemma 44, but using Lemma 45 to conclude that we read the last fulfilled write as specified by the write permission. □

Lemma 47.

$$\begin{aligned} &r = r_1 \bullet \Pi_{a \in A} r_m^a \wedge u' = u[m \mapsto [a \in A \mapsto r_m^a]] \wedge m \notin \mathit{dom}(u) \\ &\Rightarrow r_1 \bullet \mathrm{canAcq}(r_1, u') \leq_{\mathrm{o}} r \bullet \mathrm{canAcq}(r, u) \end{aligned}$$

Lemma 48.

$$\langle P, M\rangle \xrightarrow{m} \langle P', M'\rangle \wedge P(m.\mathtt{tid}, m.\mathtt{loc}) = \emptyset \Rightarrow P' = P \wedge M' = M \overset{\mathrm{A}}{\hookleftarrow} m$$

Lemma 49 (Soundness of release write)**.**

$$[\![\vdash \{x = e_1 * v = e_2 * \mathsf{Rel}(x, \phi) * \mathsf{W}^\pi(x, X) * \phi(v) * \mathsf{O}(x, v_0, t_0)\}\ [e_1]_{\mathtt{rel}} := e_2\ \{\exists t.\ \mathsf{W}^\pi(x, X \cup \{(v, t)\}) * t_0 < t\}]\!]^{\mathrm{np}}$$

where x, v, π, and X are assumed to be specification variables.

Proof. Let $(\mathit{gconf}, V, r) \in [\![x = e_1 * v = e_2 * \mathsf{Rel}(x, \phi) * \mathsf{W}^\pi(x, X) * \phi(v) * \mathsf{O}(x, v_0, t_0)]\!]^\eta_\mu(\mathcal{W})$. Since $[e_1]_{\mathtt{rel}} := e_2 \neq \textbf{skip}$ it suffices to consider the case where

$$\langle\langle(\mu, [e_1]_{\mathtt{rel}} := e_2), V\rangle, \mathit{gconf}\rangle \xrightarrow{\mathrm{NP}}_i \langle\langle\sigma', V'\rangle, \mathit{gconf}'\rangle,$$

and $\mathit{gconf} \in \lfloor r_F[i \mapsto r \bullet f], u, \mathcal{W}\rfloor_{\{i\}}$.

By Thread: Write it follows that $\mathit{gconf}' = (P', M')$, $\sigma' = (\mu, \textbf{skip})$, and

$$\langle P, M\rangle \xrightarrow{m} \langle P', M'\rangle \qquad V.\mathtt{rlx}(\eta(x)) < t \qquad V' = V \sqcup [\mathtt{pln} : \{\eta(x)@t\}, \mathtt{rlx} : \{\eta(x)@t\}]$$

and $V' = R$, $P(i, \eta(x)) = \emptyset$ where $\mathit{gconf} = (P, M)$ and $m = \langle \eta(x) :^{\mathtt{rel}}_i \eta(v), R@(\_, t]\rangle$. It follows by Lemma 48 that $P' = P$ and $M' = M \overset{\mathrm{A}}{\hookleftarrow} m$.

From the definition of separating conjunction there exists r_1 and r_2 such that $r = r_1 \bullet r_2$ with $(\mathit{gconf}, V, r_1) \in [\![\mathsf{Rel}(x, \phi) * \mathsf{W}_\pi(x, X)]\!]^\eta_\mu(\mathcal{W})$ and $(\mathit{gconf}, V, r_2) \in [\![\phi(a)]\!]^\eta_\mu(\mathcal{W})$. It follows that there exists an $\iota \in \mathit{dom}(\mathcal{W}.\mathtt{rel})$ and π' such that $\vdash \phi \Rightarrow \mathcal{W}.\mathtt{rel}(\iota)$ and $r_1.\mathtt{wr}(\eta(x)) = (\pi', \eta(X))$ with $\pi \leq \pi'$.

Let t denote $r \bullet \Pi_{t \in \mathit{TId} \setminus \{i\}} r_F(t) \bullet \Pi_{m \in M} \Pi_{\iota \in \mathit{dom}(u(m))} u(m)(\iota)$. From the definition of erasure it follows that $\vdash \mathcal{W}.\mathtt{rel}(\iota)(\eta(v)) \Rightarrow \circledast_{\iota \in t.\mathtt{acq}(\eta(x))} \mathcal{W}.\mathtt{acq}(\iota)(\eta(v))$ and thus that there exists $(r_\iota)_{\iota \in t.\mathtt{acq}(\eta(x))}$ such that $r_2 = \Pi_{\iota \in t.\mathtt{acq}(\eta(x))} r_\iota$ and $(\mathit{gconf}, V, r_\iota) \in [\![\mathcal{W}.\mathtt{acq}(\iota)(\eta(v))]\!]^{[]}_{[]}(\mathcal{W})$ for all $\iota \in t.\mathtt{acq}(\eta(x))$.

Pick $u' = u[m \mapsto (\lambda \iota \in t.\mathtt{acq}(\eta(x)).\ r_\iota)]$, $r' = r_1.\mathtt{wr}[\eta(x) \mapsto (\pi', \{(\eta(v), t)\} \cup \eta(X))]$ and $\mathcal{W}' = \mathcal{W}$. Then it remains to show that

1. $\mathit{gconf}' \in \lfloor r_F[i \mapsto r' \bullet f], u', \mathcal{W}'\rfloor_{\{i\}}$,
2. $(\mathit{gconf}', V', r') \in [\![\exists t.\ \mathcal{W}_\pi(x, X \cup \{(v, t)\}) * t_0 < t]\!]^\eta_{\sigma'.\mu}(\mathcal{W}')$,
3. $r' \bullet \mathrm{canAcq}(r', u') \leq_{\mathrm{o}} r \bullet \mathrm{canAcq}(r, u)$,
4. and $\mathrm{writesAllowed}(r, \mathrm{written}(\mathit{gconf}, \mathit{gconf}'))$.

Starting with the last proof obligation, writesAllowed$(r, \text{written}(\mathit{gconf}, \mathit{gconf}'))$: since written$(\mathit{gconf}, \mathit{gconf}') = \{m\}$, the writesAllowed$(r, \text{written}(\mathit{gconf}, \mathit{gconf}'))$ proof obligation reduces to proving that writeAllowed(r, m). This is easily seen to hold since $\pi_1(r.\mathtt{wr}(\eta(x))) > 0$.

The second-to-last proof obligation, $r' \bullet \text{canAcq}(r', u') \leq_{\text{o}} r \bullet \text{canAcq}(r, u)$ follows from Lemma 47.

The second proof obligation follows easily by the assertion semantics since $t_0 \leq V.\mathtt{rlx}(\eta(x)) < t$.

It remains to prove that $\mathit{gconf}' \in \lfloor r_F[i \mapsto r' \bullet f], u', \mathcal{W}' \rfloor_{\{i\}}$: Since $M' = M \Leftarrow m$ it follows that $M'(\eta(x)) \setminus P = (M(\eta(x)) \setminus P) \cup \{m\}$. Furthermore, since $V \leq R$ and $\mathit{gconf} \leq \mathit{gconf}'$ it also follows that

$$\forall \iota \in \mathit{dom}(u'(m)).\ (\mathit{gconf}', R, u'(m)(\iota)) \in [\![\mathcal{W}.\mathtt{acq}(\iota)(m.\mathtt{val})]\!]^{[]}_{[]}(\mathcal{W}')$$

as required, by upwards-closure. □

4.4 Rules for relaxed accesses

Lemma 50.

$$\iota \in \mathit{dom}(\mathcal{W}.\mathtt{acq}) \cap r.\mathtt{acq}(m.\mathtt{loc}) \wedge \mathit{gconf} \in \lfloor r_F[i \mapsto r \bullet f], u, \mathcal{W} \rfloor_{\{i\}} \wedge m \in \mathit{gconf}.M \Rightarrow$$
$$(\mathit{gconf}_\bot, V_\bot, \varepsilon) \in [\![\nabla(\mathcal{W}.\mathtt{acq}(\iota)(m.\mathtt{val}))]\!]^{[]}_{[]}(\mathcal{W})$$

Lemma 51 (Soundness of relaxed read)**.**

$$[\![\vdash \{x = e * \mathsf{Acq}(x, \phi) * \mathsf{O}(x, \_, t)\}\ a := [e]_{\mathtt{rlx}}\ \{\mathsf{Acq}(x, \phi) * \exists t' \geq t.\ \nabla(\phi(a)) * \mathsf{O}(x, a, t')\}]\!]^{\mathrm{np}}$$

where x and t are specification variables.

Proof. Let $(\mathit{gconf}, V, r) \in [\![x = e * \mathsf{Acq}(x, \phi) * \mathsf{O}(x, \_, t)]\!]^{\eta}_{\mu}(\mathcal{W})$. It suffices to consider the case where

$$\langle\langle(\mu, a := [e]_{\mathtt{rlx}}), V\rangle, \mathit{gconf}\rangle \xrightarrow{\mathrm{NP}}_i \langle\langle\sigma', V'\rangle, \mathit{gconf}'\rangle,$$

and $\mathit{gconf} \in \lfloor r_F[i \mapsto r \bullet f], u, \mathcal{W}' \rfloor_{\{i\}}$, $\mathcal{W}' \geq \mathcal{W}$.

By the Thread: Read rule it follows that there exists an $m = \langle \eta(x) :^o_j v, V_m @ (\_, t_m]\rangle \in M$ such that $V.\mathtt{rlx}(\eta(x)) \leq t_m$, $\mathit{gconf}'.P = \mathit{gconf}.P$, $\mathit{gconf}'.M = \mathit{gconf}.M$, $V' = V \sqcup [\mathtt{pln} : \{\eta(x)@t_m\}, \mathtt{rlx} : \{\eta(x)@t_m\}]$ and $\sigma' = (\mu[a \mapsto v], \mathbf{skip})$.

From the $(\mathit{gconf}, V, r) \in [\![\mathsf{Acq}(x, \phi) * \mathsf{O}(x, \_, t)]\!]^{\eta}_{\mu}(\mathcal{W})$ assumption it follows that there exists an $\iota \in \mathit{dom}(\mathcal{W}.\mathtt{acq}) \cap r.\mathtt{acq}(\eta(x))$ such that $\mathcal{W}.\mathtt{acq}(\iota) = \phi$ and $\eta(t) \leq V.\mathtt{rlx}(\eta(x)) \leq t_m$. Hence, by Lemma 50,

$$(\mathit{gconf}_\bot, V_\bot, \varepsilon) \in [\![\nabla(\phi(v))]\!]^{[]}_{[]}(\mathcal{W}')$$

Pick $r' = r$, $u' = u$ and $\mathcal{W}'' = \mathcal{W}'$. It thus remains to prove that

$$(\mathit{gconf}', V', r') \in [\![\mathsf{Acq}(x, \phi) * \exists t' \geq t.\ \nabla(\phi(a)) * \mathsf{O}(x, a, t')]\!]^{\eta}_{\sigma'.\mu}(\mathcal{W}'')$$

which follows easily from the definition of the interpretation. □

Lemma 52 (Soundness of relaxed read for an exclusive writer)**.**

$$[\![\vdash \{x = e * \mathsf{Acq}(x, \phi) * \mathsf{W}^1(x, X)\}\ a := [e]_{\mathtt{rlx}}\ \{B\}]\!]^{\mathrm{np}}$$

where x and X are specification variables and B is the following assertion

$$\mathsf{Acq}(x, \phi) * \mathsf{W}^1(x, X) * (\nabla\phi(a)) * \mathsf{O}(x, a, \mathit{snd}(\max(X))) * a = \mathit{fst}(\max(X)).$$

Proof. Similar to the proof of Lemma 51, but using Lemma 45 to prove properties about the value read. □

Lemma 53.

$$m.\mathtt{mod} = \mathtt{rlx} \wedge \langle P, M\rangle \xrightarrow{m} \langle P', M'\rangle \Rightarrow$$
$$\mathit{values}(M'(\mathtt{rlx})) = \mathit{values}(M(\mathtt{rlx})) \cup \{(m.\mathtt{loc}, m.\mathtt{val})\}$$

where $\mathit{values}(M) = \{(m.\mathtt{loc}, m.\mathtt{val}) \mid m \in M\}$.

Lemma 54.

$$(M, P) \in \lfloor r_F[i \mapsto r \bullet f], u, \mathcal{W} \rfloor_{\{i\}} \wedge \langle P, M\rangle \xrightarrow{m} \langle P', M'\rangle \wedge m.\mathtt{time} > \mathit{snd}(\max(X)) \wedge$$
$$m.\mathtt{mod} = \mathtt{rlx} \wedge r.\mathtt{wr}(m.\mathtt{loc}) = (\pi, X) \wedge r' = r.\mathtt{wr}[m.\mathtt{loc} \mapsto (\pi, X \cup \{(m.\mathtt{val}, m.\mathtt{time})\})] \wedge$$
$$(\mathit{gconf}_\bot, V_\bot, \varepsilon) \in [\![\mathcal{W}.\mathtt{rel}(m.\mathtt{loc})(m.\mathtt{val})]\!]^{[]}_{[]}(|\mathcal{W}|) \Rightarrow$$
$$(M', P') \in \lfloor r_F[i \mapsto r' \bullet f], u, \mathcal{W} \rfloor_{\{i\}}$$

Lemma 55 (Soundness of relaxed write)**.**

$$[\![\vdash \{A\}\ [e_1]_{\mathtt{rlx}} := e_2\ \{\exists t.\ \mathsf{W}^\pi(x, X \cup \{(v,t)\}) * t_0 < t\}]\!]^{\mathrm{np}}$$

where x, v, X and π are specification variables and A is the following assertion

$$x = e_1 * v = e_2 * \mathsf{Rel}(x,\phi) * \mathsf{W}^\pi(x,X) * \phi(v) * \mathrm{pure}(\phi(v)) * \mathsf{O}(x, v_0, t_0).$$

Proof. Let $(gconf, V, r) \in [\![A]\!]^\eta_\mu(\mathcal{W})$. It suffices to consider the case where

$$\langle\langle(\mu, [e_1]_{\mathtt{rlx}} := e_2), V\rangle, gconf\rangle \xrightarrow{\mathrm{NP}}_i \langle\langle\sigma', V'\rangle, gconf'\rangle,$$

and $gconf \in \lfloor r_F[i \mapsto r \bullet f], u, \mathcal{W}'\rfloor_{\{i\}}$, $\mathcal{W}' \geq \mathcal{W}$.

By Thread: Write it follows that $gconf' = (P', M', O)$, $\sigma' = (\mu, \mathbf{skip})$, and

$$\langle P, M\rangle \xrightarrow{m} \langle P', M'\rangle \qquad V.\mathtt{rlx}(\eta(x)) < t \qquad V' = V \sqcup [\mathtt{pln} : \{\eta(x)@t\}, \mathtt{rlx} : \{\eta(x)@t\}]$$

and $R = [\mathtt{pln} : \{\eta(x)@t\}, \mathtt{rlx} : \{\eta(x)@t\}]$, where $gconf = (P, M, O)$ and $m = \langle \eta(x) :^{\mathtt{rlx}}_i \eta(v), R@(\_, t]\rangle$.

Pick $r' = r[\mathtt{wr}(\eta(x)) \mapsto (\pi, \{(\eta(v), t)\} \cup \eta(X))]$, $u' = u$ and $\mathcal{W}'' = \mathcal{W}'$. Then we need to prove that $r' \bullet \mathrm{canAcq}(r', u) \leq_{\mathrm{o}} r \bullet \mathrm{canAcq}(r, u)$,

$$gconf' \in \lfloor r_F[i \mapsto r' \bullet f], u, \mathcal{W}'\rfloor_{\{i\}} \quad (gconf', V', r') \in [\![\exists t.\ \mathsf{W}^\pi(x, X \cup \{(v,t)\}) * \mathit{snd}(\max(X)) < t]\!]^\eta_\mu(\mathcal{W}')$$

and $\mathrm{writesAllowed}(r, \mathrm{written}(gconf, gconf'))$.

Firstly, $r' \bullet \mathrm{canAcq}(r', u) \leq_{\mathrm{o}} r \bullet \mathrm{canAcq}(r, u)$ holds trivially since $r' \leq_{\mathrm{o}} r$ and $r'.\mathtt{acq} = r.\mathtt{acq}$. Furthermore, $\mathrm{writesAllowed}(r, \mathrm{written}(gconf, gconf'))$ follows from $\mathrm{writeAllowed}(r, m)$, which itself follows from the assumption that

$$(gconf, V, r) \in [\![A]\!]^\eta_\mu(\mathcal{W}).$$

The third proof obligation, $(gconf', V', r') \in [\![\exists t.\ \mathsf{W}^\pi(x, X \cup \{(v,t)\}) * t_0 < t]\!]^\eta_\mu(\mathcal{W}')$, is easily seen to hold by the definition of the interpretation.

It remains to prove that $gconf' \in \lfloor r_F[i \mapsto r' \bullet f], u, \mathcal{W}'\rfloor_{\{i\}}$, which reduces to proving that

$$(gconf_\perp, V_\perp, \varepsilon) \in [\![\mathcal{W}.\mathtt{rel}(m.\mathtt{loc})(m.\mathtt{val})]\!]^{[]}_{[]}(|\mathcal{W}|)$$

by Lemma 54. This in turn follows from the fact that $\phi(m.\mathtt{val})$ is pure. □

4.5 Rules for plain accesses

Lemma 56 (Soundness of plain read)**.**

$$\left[\!\left[\vdash \{x = e * x \overset{\pi}{\mapsto} v\}\ a := [e]_{\mathtt{pln}}\ \{x \overset{\pi}{\mapsto} v * a = v\}\right]\!\right]^{\mathrm{np}}$$

where x and v are specification variables.

Proof. Let $(gconf, V, r) \in \left[\!\left[x = e * x \overset{\pi}{\mapsto} v\right]\!\right]^{\eta}_{\mu}(\mathcal{W})$. It suffices to consider the case where

$$\langle\langle(\mu, a := [e]_{\mathtt{pln}}), V\rangle, gconf\rangle \xrightarrow{\mathrm{NP}}_i \langle\langle\sigma', V'\rangle, gconf'\rangle,$$

and $gconf \in \lfloor r_F[i \mapsto r \bullet f], u, \mathcal{W}'\rfloor_{\{i\}}$, $\mathcal{W}' \geq \mathcal{W}$.

By the Thread: Read rule it follows that there exists an $m = \langle \eta(x) :^{\mathtt{pln}}_j v', V_m@(\_, t]\rangle \in M$ such that $V.\mathtt{pln}(\eta(x)) \leq t$, $gconf'.P = gconf.P$, $gconf'.M = gconf.M$, $V' = V \sqcup [\mathtt{pln} : \bot, \mathtt{rlx} : \{\eta(x)@t\}]$ and $\sigma' = (\mu[a \mapsto v'], \mathbf{skip})$.

From the $(gconf, V, r) \in \left[\!\left[x \overset{\pi}{\mapsto} v\right]\!\right]^{\eta}_{\mu}(\mathcal{W})$ assumption it follows that there exists a $\pi' \geq \pi$ and $t' \leq V(\eta(x)).\mathtt{pln}$ such that $r.\mathtt{pln}(\eta(x)) = inj_2(\pi', \eta(v), t')$ and $\pi' > 0$. Hence, $t' \leq t$.

From the erasure assumption it follows that $last(M(\eta(x)) \setminus P).\mathtt{val} = v$ and $last(M(\eta(x)) \setminus P).\mathtt{time} = t'$.

If $m \in P$ it follows that $m.\mathtt{tid} \neq i$ since otherwise, $V'.\mathtt{rlx}(\eta(x)) = V.\mathtt{rlx}(\eta(x)) \sqcup t < t$ which is a contradiction. Hence, by erasure,

$$\mathrm{writeAllowed}(r_F(m.\mathtt{tid}) \bullet \mathrm{canAcq}(r_F(m.\mathtt{tid}), u), m).$$

This is a contradiction, since $\mathrm{canAcq}(r_F(m.\mathtt{tid}), u).\mathtt{pln}(\eta(x)) = (1, \_)$, $r.\mathtt{pln}(\eta(x)) = (\pi, \_)$ and $r\#\mathrm{canAcq}(r_F(m.\mathtt{tid}), u)$.

Otherwise, if $m \notin P$ it follows that $m \in (M(\eta(x)) \setminus P)$ and since $t' \leq t$ it must be the case that $last(M(\eta(x)) \setminus P) = m$. It follows that $v = v'$.

Pick $r' = r$, $u' = u$ and $\mathcal{W}'' = \mathcal{W}'$. Now the remain proof obligations are easy. □

Lemma 57.

$$x \in PLoc \wedge m = \langle x :^{\mathtt{pln}}_i v, R@(\_, t]\rangle \wedge (P, M) \xrightarrow{m} (P', M') \wedge$$
$$last(M(x) \setminus P).\mathtt{time} < t \Rightarrow last(M'(x) \setminus P') = m$$

Lemma 58 (Soundness of plain write)**.**

$$\left[\!\left[\vdash \{x = e * x \overset{1}{\mapsto} \_\}\ [e]_{\mathtt{pln}} := a\ \{x \overset{1}{\mapsto} a\}\right]\!\right]^{\mathrm{np}}$$

where x is a specification variable.

Proof. Let $(gconf, V, r) \in \left[\!\left[x = e * x \overset{1}{\mapsto} \_\right]\!\right]^{\eta}_{\mu}(\mathcal{W})$. It suffices to consider the case where

$$\langle\langle(\mu, [e]_{\mathtt{pln}} := a), V\rangle, gconf\rangle \xrightarrow{\mathrm{NP}}_i \langle\langle\sigma', V'\rangle, gconf'\rangle,$$

and $gconf \in \lfloor r_F[i \mapsto r \bullet f], u, \mathcal{W}'\rfloor_{\{i\}}$, $\mathcal{W}' \geq \mathcal{W}$.

By the Thread: Write rule it follows that there exists a t such that

$$(gconf.P, gconf.M) \xrightarrow{\langle \eta(x) :^{\mathtt{pln}}_i \mu(a), R@(\_, t]\rangle} (gconf'.P, gconf'.M) \quad V' = V \sqcup [\mathtt{pln} : \{\eta(x)@t\}, \mathtt{rlx} : \{\eta(x)@t\}$$

$V(\eta(x)).\mathtt{rlx} < t$ and $\sigma' = (\mu, \mathbf{skip})$.

From the $(gconf, V, r) \in \left[\!\left[x \overset{1}{\mapsto} \_\right]\!\right]^{\eta}_{\mu}(\mathcal{W})$ assumption it follows that there exists a $t' \leq V.\mathtt{pln}(\eta(x))$ such that $r.\mathtt{pln}(\eta(x)) = inj_2(1, \_, t')$.

By definition of a timemap, $V.\mathtt{pln}(\eta(x)) \leq V.\mathtt{rlx}(\eta(x))$ and thus $t' < t$.

Pick $r' = r[\mathtt{pln}(x) \mapsto inj_2(1, \mu(a), t)]$, $\mathcal{W}' = \mathcal{W}$ and $u' = u$. By Lemma 57 it follows that $last(gconf'.M(\eta(x)) \setminus gconf'.P) = \langle \eta(x) :^{\mathtt{pln}}_{i} v, R@(\_, t] \rangle$ and thus, $gconf' \in \lfloor r_F[i \mapsto r' \bullet f], u', \mathcal{W}' \rfloor_{\{i\}}$. It remains to prove that $(gconf', V', r') \in \left[\!\left[x \overset{1}{\mapsto} a\right]\!\right]^{\eta}_{\sigma'.\mu}(\mathcal{W}')$, $r' \bullet \mathrm{canAcq}(r', u') \leq_{\mathrm{o}} r \bullet \mathrm{canAcq}(r, u)$, and $\mathrm{writesAllowed}(r, \mathrm{written}(gconf, gconf'))$. The first obligation follows easily from the definition of the interpretation. The second obligation follows easily from the fact that $r' \leq_{\mathrm{o}} r$ and $r'.\mathtt{acq} = r.\mathtt{acq}$. For the last obligation, it suffices to prove that $\mathrm{writeAllowed}(r, \langle \eta(x) :^{\mathtt{pln}}_{i} v, R@(\_, t] \rangle)$, which holds since $r.\mathtt{pln}(\eta(x)) = inj_2(1, \_)$ by assumption. □

5 Examples

5.1 Random number generator

We do not need to transfer any information, so we can pick $\lambda v.\ \top$ as the predicate for x and y.

$$x = y = 0$$

$$\begin{array}{l|l}
\{\mathsf{W}^1(y,[0]) * \mathsf{Rel}(y,\lambda v.\ \top) * \mathsf{Acq}(x,\lambda v.\ \top) * \mathsf{O}(x,0,0)\} & \{\mathsf{W}^1(x,[0]) * \mathsf{Rel}(x,\lambda v.\ \top) * \mathsf{Acq}(y,\lambda v.\ \top) * \mathsf{O}(y,0,0)\} \\
\quad \{x = x * \mathsf{Acq}(x,\lambda v.\ \top) * \mathsf{O}(x,0,0)\} & \quad \{y = y * \mathsf{Acq}(y,\lambda v.\ \top) * \mathsf{O}(y,0,0)\} \\
r_1 := [x]_{\mathsf{rlx}}; & r_2 := [y]_{\mathsf{rlx}}; \\
\quad \{\mathsf{Acq}(x,\lambda v.\ \top) * (\exists t_1.\ \mathsf{O}(x,r_1,t_1) * 0 \le t_1) * \nabla\top\} & \quad \{\mathsf{Acq}(y,\lambda v.\ \top) * (\exists t_1.\ \mathsf{O}(x,r_2,t_1) * 0 \le t_1) * \nabla\top\} \\
\quad \{\mathsf{O}(x,r_1,_)\} & \quad \{\mathsf{O}(y,r_2,_)\} \\
\{\mathsf{W}^1(y,[0]) * \mathsf{Rel}(y,\lambda v.\ \top) * \mathsf{O}(x,r_1,_)\} & \{\mathsf{W}^1(x,[0]) * \mathsf{Rel}(x,\lambda v.\ \top) * \mathsf{O}(y,r_2,_)\} \\
\quad \{\mathsf{W}^1(y,[0]) * \mathsf{Rel}(y,\lambda v.\ \top) * \top * \mathrm{pure}(\top)\} & \quad \{\mathsf{W}^1(x,[0]) * \mathsf{Rel}(x,\lambda v.\ \top) * \top * \mathrm{pure}(\top)\} \\
[y]_{\mathsf{rlx}} := r_1 + 1 & [x]_{\mathsf{rlx}} := r_2 \\
\quad \{\mathsf{W}^1(y,[r_1+1;0]) * \mathsf{Rel}(y,\lambda v.\ \top)\} & \quad \{\mathsf{W}^1(x,[r_2;0]) * \mathsf{Rel}(x,\lambda v.\ \top)\} \\
\quad \{\mathsf{W}^1(y,[r_1+1;0])\} & \quad \{\mathsf{W}^1(x,[r_2;0])\} \\
\{\mathsf{W}^1(y,[r_1+1;0]) * \mathsf{O}(x,r_1,_)\} & \{\mathsf{W}^1(x,[r_2;0]) * \mathsf{O}(y,r_2,_)\}
\end{array}$$

$$\{r_1 \in \{r_2, 0\} \wedge r_2 \in \{r_1 + 1, 0\}\}$$
$$\{r_1 = 0 \wedge (r_2 = 0 \vee r_2 = 1)\}$$

5.2 Separation

We can define a self-looping list by having a location hold the address of the next element of the list, and using an address already in the list to mark the end the list:

$$\begin{aligned}
\mathit{is\text{-}list}(v) &\stackrel{\mathit{def}}{=} \exists s_2.\ \mathit{finite}(s_2) * v \notin s_2 * \mathit{is\text{-}list}'(v, \{v\}, s_2) \\
\mathit{is\text{-}list}'(v, s_1, s_2) &\stackrel{\mathit{def}}{=} \exists \ell, v'.\ \mathsf{W}^1(v, v' :: \ell) * \mathsf{Rel}(v, \lambda v.\ \top) * \\
&\qquad (v' \in s_1 \vee (v' \in s_2 * \mathit{is\text{-}list}'(v', s_1 \cup \{v'\}, s_2 \setminus \{v'\})))
\end{aligned}$$

Because we do not transfer anything between threads (we are trying to show the threads do not interact!), we can use $\lambda v.\ \top$ as the the predicate for the locations of the list. To avoid clutter, we elide $\mathsf{Rel}(-, \lambda v.\ \top)$ and the associated assertions below.

We then have the following proof outline for the first thread (the second thread is entirely symmetric):

$$\begin{array}{l}
\{\mathit{is\text{-}list}(a)\} \\
\{\exists s.\ \mathit{finite}(s) * a \notin s * \mathit{is\text{-}list}'(a, \{a\}, s)\} \\
\{\exists s, a'.\ \mathsf{W}^1(a, a' :: _) * (a' \in \{a\} \vee (a' \notin \{a\} * \mathit{is\text{-}list}'(a', \{a\} \cup \{v'\}, s \setminus \{a'\})))\} \\
\quad \{a = a * \ldots * \mathsf{W}^1(a, a' :: _)\} \\
r_1 := [a]_{\mathsf{rlx}}; \\
\quad \{\ldots * \mathsf{W}^1(a, a' :: _) * \ldots * \mathsf{O}(a, r_1, _) * r_1 = a'\} \\
\{\exists s, a'.\ \mathsf{W}^1(a, a' :: _) * r_1 = a' * (a' \in \{a\} \vee (a' \notin \{a\} * \mathit{is\text{-}list}'(a', \{a\} \cup \{a'\}, s \setminus \{a'\})))\} \\
\{\exists a'.\ \mathsf{W}^1(a, a' :: _) * r_1 = a' * (a' \in \{a\} \vee (a' \notin \{a\} * \exists a''.\ \mathsf{W}^1(a', a'' :: _) * \ldots))\} \\
\quad \{r_1 = a' * a = a * \mathsf{W}^1(a', _) * \ldots\} \\
[r_1]_{\mathsf{rlx}} := a \\
\quad \{\mathsf{W}^1(a', a :: _) * \ldots\} \\
\{\exists a'.\ \mathsf{W}^1(a, a' :: _) * ((a' \in \{a\}) \vee (a' \notin \{a\} * \mathsf{W}^1(a', a :: _)))\} \\
\{\mathsf{W}^1(a, a :: _) \vee (\exists a'.\ \mathsf{W}^1(a, a' :: _) * a' \notin \{a\} * \mathsf{W}^1(a', a :: _))\} \\
\{(\mathit{is\text{-}list}'(a, \{a\}, \varnothing)) \vee (\exists a'.\ a' \notin \{a\} * \mathit{is\text{-}list}'(a, \{a\}, \{a'\}))\} \\
\{\mathit{is\text{-}list}(a)\}
\end{array}$$

5.3 Non-deterministic write

5.3.1 Example using non-deterministic choice

Invariant-based methods cannot be used to prove that $\neg(r_1 = r_2 = 1)$ in the following program where all locations initially hold 0, and where the $*$ stands for a non-deterministic choice.

$$\begin{array}{l} \textbf{if } * \ [x]_{\mathsf{rlx}} := 1 \\ \textbf{else } [y]_{\mathsf{rlx}} := 1 \end{array} \Bigg\| \begin{array}{l} r_1 := [x]_{\mathsf{rlx}}; \\ r_2 := [y]_{\mathsf{rlx}}; \end{array} \tag{nondet}$$

We elide most things.

$$\begin{array}{l} \{\mathsf{W}^1(x,[0]) * \mathsf{W}^1(y,[0])\} \\ \textbf{if } * \\ \quad [x]_{\mathsf{rlx}} := 1 \\ \quad \{\mathsf{W}^1(x,[1;0]) * \mathsf{W}^1(y,[0])\} \\ \textbf{else} \\ \quad [y]_{\mathsf{rlx}} := 1 \\ \quad \{\mathsf{W}^1(x,[0]) * \mathsf{W}^1(y,[1;0])\} \\ \{(\mathsf{W}^1(x,[1;0]) * \mathsf{W}^1(y,[0])) \vee (\mathsf{W}^1(x,[0]) * \mathsf{W}^1(y,[1;0]))\} \end{array} \Bigg\| \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \{\mathsf{Acq}(x, \lambda v.\ \top) * \mathsf{Acq}(y, \lambda v.\ \top)\} \\ r_1 := [x]_{\mathsf{rlx}}; \\ r_2 := [y]_{\mathsf{rlx}}; \\ \{\mathsf{O}(x, r_1, \_) * \mathsf{O}(y, r_2, \_)\} \end{array}$$

We can conclude, by case analysis on the disjunction, and by using lemma 37, that $(r_1 \in \{0,1\} \wedge r_2 \in \{0\}) \vee (r_1 \in \{0\} \wedge r_2 \in \{0,1\})$, and therefore $\neg(r_1 = r_2 = 1)$.

5.3.2 Example within the language

Invariant-based methods cannot be used to prove that $\neg(r_1 = r_2 = 1)$ in the following program where all locations initially hold 0.

$$[t]_{\mathsf{rlx}} := 1 \Bigg\| \begin{array}{l} r_3 := [t]_{\mathsf{rlx}}; \\ \textbf{if } r_3\ [x]_{\mathsf{rlx}} := 1 \\ \textbf{else } [y]_{\mathsf{rlx}} := 1 \end{array} \Bigg\| \begin{array}{l} r_1 := [x]_{\mathsf{rlx}}; \\ r_2 := [y]_{\mathsf{rlx}} \end{array}$$

5.4 Coherence

5.4.1 CoRW

The following program, where location x initially holds 0, cannot have the following outcome:

$$\begin{array}{l} r_1 := [x]_{\mathsf{rlx}};\ \text{// reads 2} \\ [x]_{\mathsf{rlx}} := 1 \end{array} \Bigg\| [x]_{\mathsf{rlx}} := 2 \Bigg\| \begin{array}{l} r_2 := [x]_{\mathsf{rlx}};\ \text{// reads 1} \\ r_3 := [x]_{\mathsf{rlx}}\ \text{// reads 2} \end{array} \tag{CoRW}$$

The proof outline for the first thread crucially records that the value it reads has a smaller timestamp than the value it writes:

$$\begin{array}{l} \{\mathsf{W}^{\frac{1}{2}}(x, \{(0,0)\}) * \mathsf{Acq}(x, \lambda v.\ \top)\} \\ r_1 := [x]_{\mathsf{rlx}}; \\ \{\exists t.\ \mathsf{W}^{\frac{1}{2}}(x, \{(0,0)\}) * \mathsf{O}(x, r_1, t)\} \\ [x]_{\mathsf{rlx}} := 1 \\ \{\exists t, t_1.\ \mathsf{W}^{\frac{1}{2}}(x, \{(0,0),(1,t_1)\}) * \mathsf{O}(x, r_1, t) * t < t_1\} \end{array}$$

The proof outline for the second thread just records it writes 2:

$$\begin{array}{l} \{\mathsf{W}^{\frac{1}{2}}(x, \{(0,0)\})\} \\ [x]_{\mathsf{rlx}} := 2 \\ \{\exists t_2.\ \mathsf{W}^{\frac{1}{2}}(x, \{(0,0),(2,t_2)\}) * 0 < t_2\} \end{array}$$

The proof outline for the third thread just records the order of the timestamps of 1 and 2:

$$\begin{array}{l}\{\mathsf{Acq}(x, \lambda v.\ \top)\}\\ r_2 := [x]_{\mathtt{rlx}};\\ \{\mathsf{Acq}(x, \lambda v.\ \top) * \exists t_a.\ \mathsf{O}(x, r_1, t_a)\}\\ r_3 := [x]_{\mathtt{rlx}}\\ \{\exists t_a, t_b.\ \mathsf{O}(x, r_2, t_a) * \mathsf{O}(x, r_3, t_b) * t_a \leq t_b\}\end{array}$$

We can then combine the write permissions to obtain $\mathsf{W}^1(x, \{(0,0), (1,t_1), (2,t_2)\})$. If moreover we assume that $r_2 = 1 * r_3 = 2$, from the postcondition of the third thread, we have that $t_1 < t_2$ by Lemmas 37 and 39. Also, if moreover we assume that $r_1 = 2$, we have that $t_2 < t_1$ by Lemma 37. Therefore, we can conclude $\neg(r_1 = 2 * r_2 = 1 * r_3 = 2)$.

5.4.2 CoWR

The following program, where location x initially holds 0, cannot have the following outcome:

$$\begin{array}{l}[x]_{\mathtt{rlx}} := 1;\\ r_1 := [x]_{\mathtt{rlx}}\ \text{// reads 2}\end{array} \Bigg\| [x]_{\mathtt{rlx}} := 2 \Bigg\| \begin{array}{l}r_2 := [x]_{\mathtt{rlx}};\ \text{// reads 2}\\ r_3 := [x]_{\mathtt{rlx}}\ \text{// reads 1}\end{array} \tag{CoWR}$$

The proof outline for the first thread records that the read reads a timestamp no older than that of the write:

$$\begin{array}{l}\{\mathsf{W}^{\frac{1}{2}}(x, \{(0,0)\}) * \mathsf{Acq}(x, \lambda v.\ \top)\}\\ [x]_{\mathtt{rlx}} := 1\\ \{\exists t_1.\ \mathsf{W}^{\frac{1}{2}}(x, \{(0,0), (1,t_1)\}) * \mathsf{Acq}(x, \lambda v.\ \top)\}\\ r_1 := [x]_{\mathtt{rlx}};\\ \{\exists t, t_1.\ \mathsf{W}^{\frac{1}{2}}(x, \{(0,0), (1,t_1)\}) * \mathsf{O}(x, r_1, t) * t_1 \leq t\}\end{array}$$

The proof outline for the second and the third threads are identical to that of CoRW. As for CoRW, we can combine the write permissions to obtain $\mathsf{W}^1(x, \{(0,0), (1,t_1), (2,t_2)\})$. If moreover we assume that $r_2 = 2 * r_3 = 1$, symmetrically to CoRW, from the postcondition of the third thread, we have that $t_2 < t_1$ by Lemmas 37 and 39. Also, if moreover we assume that $r_1 = 2$, we have that $t_1 \leq t_2$ by Lemma 37, and thus $t_1 < t_2$ by Lemma 39. Therefore, we can conclude $\neg(r_1 = 2 * r_2 = 2 * r_3 = 1)$.

5.5 Release/acquire

5.5.1 Split permission message passing example

The following program, where locations x, y, and z initially hold 0, cannot have the following outcome: $(r_1 = 1 * r_2 = 0) * (r_3 = 1 * r_4 = 0)$:

$$\begin{array}{l}[x]_{\mathtt{rlx}} := 1;\\ [y]_{\mathtt{rlx}} := 1;\\ [z]_{\mathtt{rel}} := 1\end{array} \Bigg\| \begin{array}{l}r_1 := [z]_{\mathtt{acq}};\\ \textbf{if } r_1 = 1\\ \quad r_2 := [x]_{\mathtt{rlx}}\end{array} \Bigg\| \begin{array}{l}r_3 := [z]_{\mathtt{acq}};\\ \textbf{if } r_3 = 1\\ \quad r_4 := [y]_{\mathtt{rlx}}\end{array} \tag{Split-MP}$$

We pick

$$\phi_z = \lambda v.\ \big(\mathit{if}\ v = 1\ \mathit{then}\ \mathsf{W}^1(x, [1;0])\ \mathit{else}\ \top\big) * \big(\mathit{if}\ v = 1\ \mathit{then}\ \mathsf{W}^1(y, [1;0])\ \mathit{else}\ \top\big)$$

for z. This ought to be $\lambda v.\ \mathit{if}\ v = 1\ \mathit{then}\ (\mathsf{W}^1(x, [1;0]) * \mathsf{W}^1(y, [1;0]))\ \mathit{else}\ \top$, but the ϕ are stored and treated as syntactic assertions.

The proof outline for the writer transfers the write permissions for x and y away on z using ϕ_z:

$$\begin{array}{l}
\{\mathsf{W}^1(x,[0]) * \mathsf{Rel}(x,\lambda v.\ \top) * \mathsf{W}^1(y,[0]) * \mathsf{Rel}(y,\lambda v.\ \top) * \mathsf{W}^1(z,[0]) * \mathsf{Rel}(z,\phi_z)\} \\
[x]_{\mathsf{rlx}} := 1; \\
\{\mathsf{W}^1(x,[1;0]) * \mathsf{W}^1(y,[0]) * \mathsf{Rel}(y,\lambda v.\ \top) * \mathsf{W}^1(z,[0]) * \mathsf{Rel}(z,\phi_z)\} \\
[y]_{\mathsf{rlx}} := 1; \\
\{\mathsf{W}^1(x,[1;0]) * \mathsf{W}^1(y,[1;0]) * \mathsf{W}^1(z,[0]) * \mathsf{Rel}(z,\phi_z)\} \\
\quad \{z = z * 1 = 1 * \mathsf{W}^1(z,\{(0,0)\}) * \mathsf{Rel}(z,\phi_z) * \phi_z(1) * \mathsf{O}(z,0,0)\} \\
[z]_{\mathsf{rel}} := 1 \\
\quad \{\exists t_1.\ \mathsf{W}^1(z,\{(1,t_1)\} \cup \{(0,0)\}) * 0 < t_1\} \\
\{\mathsf{W}^1(z,[1;0]) * \mathsf{Rel}(z,\phi_z)\}
\end{array}$$

The proof outline for the first reader starts with an acquire permission for first half of ϕ_z for z, obtained by splitting ϕ_z from the initial $\mathsf{Acq}(z,\phi_z)$ using lemma 36, and uses it to obtain $\mathsf{W}^1(x,[1;0])$, which it then uses to know that it reads 1 from x.

$$\begin{array}{l}
\{\mathsf{Acq}(z,\lambda v.\ \mathit{if}\ v=1\ \mathit{then}\ \mathsf{W}^1(x,[1;0])\ \mathit{else}\ \top)) * \mathsf{O}(z,0,0) * \mathsf{Acq}(x,\lambda v.\ \top)\} \\
\quad \{z = z * \mathsf{Acq}(z,\lambda v.\ \mathit{if}\ v=1\ \mathit{then}\ \mathsf{W}^1(x,[1;0])\ \mathit{else}\ \top)) * \mathsf{O}(z,0,0)\} \\
r_1 := [z]_{\mathsf{acq}}; \\
\quad \left\{\begin{array}{l}\mathsf{Acq}(z,(\lambda v.\ \mathit{if}\ v=1\ \mathit{then}\ \mathsf{W}^1(x,[1;0])\ \mathit{else}\ \top)[r_1 \mapsto \top]) * \\ (\exists t_1^z.\ \mathsf{O}(z,r_1,t_1^z) * 0 \le t_1^z) * (\lambda v.\ \mathit{if}\ v=1\ \mathit{then}\ \mathsf{W}^1(x,[1;0])\ \mathit{else}\ \top)(r_1)\end{array}\right\} \\
\{(\lambda v.\ \mathit{if}\ v=1\ \mathit{then}\ \mathsf{W}^1(x,[1;0])\ \mathit{else}\ \top)(r_1) * \mathsf{Acq}(x,\lambda v.\ \top)\} \\
\textbf{if}\ r_1 = 1 \\
\quad \{\mathsf{W}^1(x,[1;0]) * \mathsf{Acq}(x,\lambda v.\ \top)\} \\
\quad r_2 := [x]_{\mathsf{rlx}} \\
\quad \{\mathsf{Acq}(x,\lambda v.\ \top) * \mathsf{W}^1(x,[1;0]) * (\nabla(\lambda v.\ \top)(r_2)) * \mathsf{O}(x,r_2,\mathit{snd}(\max([1;0]))) * r_2 = \mathit{fst}(\max([1;0]))\} \\
\quad \{r_2 = 1\} \\
\{r_1 = 1 \Longrightarrow r_2 = 1\}
\end{array}$$

The proof outline for the second reader is symmetric.

5.5.2 WRC

The following program, where locations x and y initially hold 0, cannot have the following outcome: $r_2 = 1 * r_3 = 0$:

$$[x]_{\mathsf{rlx}} := 1; \left\| \begin{array}{l} r_1 := [x]_{\mathsf{acq}}; \\ \textbf{if}\ r_1 = 1 \\ \quad [y]_{\mathsf{rel}} := 1 \end{array} \right\| \begin{array}{l} r_2 := [y]_{\mathsf{acq}}; \\ \textbf{if}\ r_2 = 1 \\ \quad r_3 := [x]_{\mathsf{rlx}} \end{array} \qquad \text{(WRC)}$$

We pick $\phi_y = \lambda v.\ \mathit{if}\ v = 1\ \mathit{then}\ \exists t.\ \mathsf{O}(x,1,t)\ \mathit{else}\ \top$. The proof outline for the first thread just records the write of 1 to x:

$$\begin{array}{l}
\{\mathsf{W}^1(x,\{(0,0)\}) * \mathsf{Rel}(x,\lambda v.\ \top)\} \\
[x]_{\mathsf{rlx}} := 1; \\
\{\exists t.\ \mathsf{W}^1(x,\{(0,0),(1,t)\}) * 0 < t\}
\end{array}$$

The proof outline for the second thread feeds the $\mathsf{O}(x,1,t)$ it obtains in the 'if' branch in the release write to y:

$$\begin{array}{l}
\{\mathsf{Acq}(x,\lambda v.\ \top) * \mathsf{O}(x,0,0) * \mathsf{W}^1(y,\{(0,0)\}) * \mathsf{Rel}(y,\phi_y)\} \\
r_1 := [x]_{\mathsf{rlx}}; \\
\textbf{if}\ r_1 = 1 \\
\quad \{\exists t.\ \mathsf{O}(x,1,t) * \mathsf{W}^1(y,\{(0,0)\}) * \mathsf{Rel}(y,\phi_y)\} \\
\quad\quad \{y = y * 1 = 1 * \mathsf{Rel}(y,\phi_y) * \mathsf{W}^1(y,\{(0,0)\}) * (\exists t.\ \mathsf{O}(x,1,t)) * \mathsf{O}(y,0,0)\} \\
\quad [y]_{\mathsf{rel}} := 1 \\
\quad\quad \{\exists t'.\ \mathsf{W}^1(y,\{(0,0)\} \cup \{(1,t')\}) * 0 < t'\} \\
\quad \{\top\}
\end{array}$$

The proof outline for the third thread relates the timestamp of the value it reads from x with what it learns using ϕ_y from the acquire read from y:

$$
\begin{array}{l}
\{\mathsf{Acq}(y, \phi_y) * \mathsf{O}(y, 0, 0) * \mathsf{Acq}(x, \lambda v.\ \top)\} \\
\quad \{y = y * \mathsf{Acq}(y, \phi_y) * \mathsf{O}(y, 0, 0)\} \\
r_2 := [y]_{\mathsf{acq}}; \\
\quad \{\exists t' \geq 0.\ \mathsf{Acq}(y, \phi_y[r_2 \mapsto \top]) * \phi(r_2) * \mathsf{O}(y, r_2, t')\} \\
\{\mathsf{Acq}(x, \lambda v.\ \top) * \phi(r_2)\} \\
\textbf{if}\ r_2 = 1 \\
\quad \{\exists t.\ \mathsf{Acq}(x, \lambda v.\ \top) * \mathsf{O}(x, 1, t)\} \\
\quad r_3 := [x]_{\mathsf{rlx}} \\
\quad \{\exists t, t'.\ \mathsf{O}(x, 1, t) * \mathsf{O}(x, r_3, t') * t \leq t'\}
\end{array}
$$

At the end of the execution, assuming $r_2 = 1$, we can confront $\mathsf{W}^1(x, \{(0,0),(1,t)\})$ with $\exists t, t'.\ \mathsf{O}(x, 1, t) * \mathsf{O}(x, r_3, t') * t \leq t'$ to conclude that $r_3 = 1$, as desired.