Fictional Separation Logic: Appendix

Jonas B. Jensen Lars Birkedal

October 2011
(updated September 2013)

Contents

A-1 Additional Lemmas

$$\frac{}{\{\sigma \circ \sigma'\} \dashv\vdash \{\sigma\} * \{\sigma'\}} \textsc{Homomorphism}$$

The following definition will be used in this appendix for the sake of brevity.

Definition A-1. Given $I : \Sigma \searrow \Sigma'$ and $\phi : \Sigma$, the function $I^\phi : \mathcal{P}(\Sigma) \to \mathcal{P}(\Sigma')$ is a lifting of I to predicates. It is defined as $I^\phi(P) \triangleq \exists \sigma \in P.\ I(\sigma \circ \phi).\diamond$

The definition above can be used to giver shorter definitions of indirect triple and indirect entailment:

$$I.\ \{P\}\ C\ \{Q\} = \forall \phi.\ \{I^\phi(P)\}\ C\ \{I^\phi(Q)\}$$
$$P \models_I Q = \forall \phi.\ (I^\phi(P) \vdash I^\phi(Q))$$

These definitions are equivalent to the original ones in the main text. Notice that $h \in I^\phi(P)$ iff $\exists \sigma \in P.\ h \in I(\sigma \circ \phi)$.

A-1.1 Specification Logic Rules

Lemma A-1. *If $e : expr$ and $I : \Sigma \searrow \Sigma'$, then $I^\phi(P \wedge e) \dashv\vdash I^\phi(P) \wedge e$.*

Note first how to read the above lemma. We have $e : expr = stack \to val$. When such an expression appears in the place of an assertion in $asn(\Sigma)$, it should be read as being implicitly injected by the function

$$inj_\Sigma(e) \triangleq \lambda s : stack.\ \{\sigma : \Sigma \mid e(s) = \mathsf{true}\}.$$

This is standard in Hoare logics, but we are explicit about it here because the e above appears injected into two different assertion logics: $asn(\Sigma)$ on the left and $asn(\Sigma')$ on the right.

Proof (of Lemma A-1).

$$\begin{array}{ll}
I^\phi(P \wedge e) & \dashv\vdash \\
\exists \sigma \in (P \wedge inj_\Sigma(e)).\ I(\sigma \circ \phi) & \dashv\vdash \\
\exists \sigma.\ \sigma \in P \wedge \sigma \in inj_\Sigma(e) \wedge I(\sigma \circ \phi) & \dashv\vdash \\
\exists \sigma.\ \sigma \in P \wedge e = \mathsf{true} \wedge I(\sigma \circ \phi) & \dashv\vdash \\
(\exists \sigma.\ \sigma \in P \wedge I(\sigma \circ \phi)) \wedge e = \mathsf{true} & \dashv\vdash \\
(\exists \sigma \in P.\ I(\sigma \circ \phi)) \wedge inj_{\Sigma'}(e) & \dashv\vdash \\
I^\phi(P) \wedge e & \square
\end{array}$$

In the following proofs, we omit $(S \vdash)$ on each line.

Proof of control flow commands.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{I.\ \{P \wedge e\}\ C\ \{P\}}{\forall \phi.\ \{I^\phi(P \wedge e)\}\ C\ \{I^\phi(P)\}}\ \text{(definition)}}{\forall \phi.\ \{I^\phi(P) \wedge e\}\ C\ \{I^\phi(P)\}}\ \text{Lemma A-1}}{\forall \phi.\ \{I^\phi(P)\}\ \mathsf{while}\ e\ \mathsf{do}\ C\ \{I^\phi(P) \wedge \neg e\}}\ \textsc{While-std}}{\forall \phi.\ \{I^\phi(P)\}\ \mathsf{while}\ e\ \mathsf{do}\ C\ \{I^\phi(P \wedge \neg e)\}}\ \text{Lemma A-1}}{I.\ \{P\}\ \mathsf{while}\ e\ \mathsf{do}\ C\ \{P \wedge \neg e\}}\ \text{(definition)}$$

Proofs of (if, call and $(;)$) have the same structure.

Proof of assign.

$$\dfrac{\dfrac{\dfrac{}{\forall \phi.\ \{I^\phi(Q)[e/x]\}\ x := e\ \{I^\phi(Q)\}}\ \textsc{Assign-std}}{\forall \phi.\ \{I^\phi(Q[e/x])\}\ x := e\ \{I^\phi(Q)\}}\ \text{(substitution on lifting)}}{I.\ \{Q[e/x]\}\ x := e\ \{Q\}}\ \text{(definition)}$$

Proof of the rule of consequence. This proof works simply by expanding the definition of the indirect triple and appealing to the standard rule of consequence:

$$\dfrac{P \vdash P' \qquad \dfrac{I.\ \{P'\}\ C\ \{Q'\}}{\forall \phi.\ \{\exists \sigma \in P'.\ I(\sigma \circ \phi)\}\ C\ \{\exists \sigma \in Q'.\ I(\sigma \circ \phi)\}} \qquad Q' \vdash Q}{\dfrac{\forall \phi.\ \{\exists \sigma \in P.\ I(\sigma \circ \phi)\}\ C\ \{\exists \sigma \in Q.\ I(\sigma \circ \phi)\}}{I.\ \{P\}\ C\ \{Q\}}}$$

Proof of the existential rule.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\forall x : A.\ I.\ \{P(x)\}\ C\ \{Q\}}{\forall x : A.\ \forall \phi.\ \{I^\phi(P(x))\}\ C\ \{I^\phi(Q)\}}\ \text{(definition)}}{\forall \phi.\ \forall x : A.\ \{I^\phi(P(x))\}\ C\ \{I^\phi(Q)\}}}{\forall \phi.\ \{\exists x : A.\ I^\phi(P(x))\}\ C\ \{I^\phi(Q)\}}\ \textsc{Exists-std}}{\forall \phi.\ \{\exists x : A.\ \exists \sigma \in P(x).\ I(\sigma \circ \phi)\}\ C\ \{I^\phi(Q)\}}\ \text{(definition)}}{\forall \phi.\ \{\exists \sigma.\ \exists x : A.\ \sigma \in P(x) \wedge I(\sigma \circ \phi)\}\ C\ \{I^\phi(Q)\}}\ \text{(def. of } (\exists) \text{ in } asn)}{\forall \phi.\ \{\exists \sigma.\ \sigma \in (\exists x \in A.\ P(x)) \wedge I(\sigma \circ \phi)\}\ C\ \{I^\phi(Q)\}}\ \text{(definition)}}{\forall \phi.\ \{I^\phi(\exists x : A.\ P(x))\}\ C\ \{I^\phi(Q)\}}\ \text{(definition)}}{I.\ \{\exists x : A.\ P(x)\}\ C\ \{Q\}}\ \text{(definition)}$$

This rule is often presented in a more symmetric-looking style, where the conclusion has an existential in both the pre- and postcondition. The symmetric version of the rule can be derived from this one by the rule of consequence.

Proof of the frame rule.

$$\dfrac{\dfrac{mods(C) \# fv(R)}{\forall \sigma_2.\ mods(C) \# fv(\sigma_2 \in R)} \quad \dfrac{\dfrac{I.\ \{P\}\ C\ \{Q\}}{\forall \phi.\ \{I^{\phi}(P)\}\ C\ \{I^{\phi}(Q)\}}\ \text{(definition)}}{\forall \phi', \sigma_2.\ \{I^{\sigma_2 \circ \phi'}(P)\}\ C\ \{I^{\sigma_2 \circ \phi'}(Q)\}}\ \text{(specialize)}}{\dfrac{\forall \phi', \sigma_2.\ \{\sigma_2 \in R \wedge I^{\sigma_2 \circ \phi'}(P)\}\ C\ \{\sigma_2 \in R \wedge I^{\sigma_2 \circ \phi'}(Q)\}}{\dfrac{\forall \phi', \sigma_2.\ \left\{\begin{array}{l}\sigma_2 \in R \wedge \exists \sigma_1 \in P. \\ I(\sigma_1 \circ \sigma_2 \circ \phi')\end{array}\right\}\ C\ \left\{\begin{array}{l}\sigma_2 \in R \wedge \exists \sigma_1 \in Q. \\ I(\sigma_1 \circ \sigma_2 \circ \phi')\end{array}\right\}}{\dfrac{\forall \phi'.\ \left\{\begin{array}{l}\exists \sigma_2 \in R.\ \exists \sigma_1 \in P. \\ I(\sigma_1 \circ \sigma_2 \circ \phi')\end{array}\right\}\ C\ \left\{\begin{array}{l}\exists \sigma_2 \in R.\ \exists \sigma_1 \in Q. \\ I(\sigma_1 \circ \sigma_2 \circ \phi')\end{array}\right\}}{\dfrac{\forall \phi'.\ \{\exists \sigma \in (P * R).\ I(\sigma \circ \phi')\}\ C\ \{\exists \sigma \in (Q * R).\ I(\sigma \circ \phi')\}}{I.\ \{P * R\}\ C\ \{Q * R\}}\ \text{(definition)}}\ \text{(definition)}}\ \textsc{Exists-std}}\ \text{(definition)}}\ \textsc{Frame-std}$$

Notice that in the place where we appeal to the standard frame rule, we are framing out a *pure* fact conjoined with a non-pure fact. The frame rule applies here because $p \wedge Q \dashv\vdash (p \wedge emp) * Q$ when p is pure.

Proof of Basic.

$$\begin{array}{c} 1.\ \{P\}\ C\ \{Q\} \\ \hline\hline \forall \phi.\ \{\exists \sigma \in P.\ 1(\sigma \circ \phi)\}\ C\ \{\exists \sigma \in Q.\ 1(\sigma \circ \phi)\} \quad \text{(definition)} \\ \hline\hline \forall \phi.\ \{\exists \sigma \in P.\ \{\sigma \circ \phi\}\}\ C\ \{\exists \sigma \in Q.\ \{\sigma \circ \phi\}\} \quad \text{(definition)} \\ \hline\hline \forall \phi.\ \{\exists \sigma \in P.\ \{\sigma\} * \{\phi\}\}\ C\ \{\exists \sigma \in Q.\ \{\sigma\} * \{\phi\}\} \quad \textsc{Homomorphism} \\ \hline\hline \{\exists \sigma \in P.\ \{\sigma\}\}\ C\ \{\exists \sigma \in Q.\ \{\sigma\}\} \quad \star \\ \hline\hline \{P\}\ C\ \{Q\} \end{array}$$

The proof step labelled ($\star$) uses the frame rule in the bottom-to-top direction and specializes ϕ to 0 in the top-to-bottom direction.

A-1.2 Separating Products

$$\dfrac{S \vdash \forall \phi.\ I * 1.\ \{P \times J(\sigma \circ \phi)\}\ C\ \{Q \times J(\sigma' \circ \phi)\}}{S \vdash I * J.\ \{P \times \{\sigma\}\}\ C\ \{Q \times \{\sigma'\}\}}\ \textsc{Product1}$$

(double line)

$$\overline{(P * Q) \times R \dashv\vdash P^{\mathsf{L}} * (Q \times R)} \qquad \overline{P \times Q \dashv\models_{1_\Sigma * 1_\Sigma} (P * Q)^{\mathsf{L}}}$$

On assertions, $\times$ distributes over $\forall$ and $\exists$ (inhabited types only). Nothing happens at $\top$, but $\bot \times R \dashv\vdash \bot$. Implication is $(P \Rightarrow Q) \times R \vdash P \times R \Rightarrow Q \times R$.

A-1.3 Pointwise Interpretations

Proof of Lemma 1.

a. The side condition ensures that $supp(f \circ g) = supp(f) \uplus supp(g)$, so the iterated separating conjunction can be split up in those two parts.

b. The side condition and the separating conjunction together entail that the supports of f and g are disjoint, and then case (a) applies.

c. We will use the fact that for any $f : X \overset{\text{fin}}{\Rightarrow} \Sigma$, $x : X$ and $\sigma : \Sigma$, we have $[x \mapsto \sigma] \circ f = [x \mapsto \sigma \circ f(x)] \circ f[x \mapsto 0]$, where $f[x \mapsto 0]$ denotes the function that maps x to 0 and any other x' to $f(x')$. Note that both the composition in Σ and in $X \overset{\text{fin}}{\Rightarrow} \Sigma$ are written as $(\circ)$. We derive the lemma as follows, where $(S \vdash)$ is omitted on each line.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\forall \phi.\ \{I([x \mapsto \sigma \circ \phi]) \wedge p\}\ C\ \{I([x \mapsto \sigma' \circ \phi]) \wedge q\}}{\forall f.\ \{I([x \mapsto \sigma \circ f(x)]) \wedge p\}\ C\ \{I([x \mapsto \sigma' \circ f(x)]) \wedge q\}}\ \text{(renaming)}}{\forall f.\ \left\{\begin{matrix} I([x \mapsto \sigma \circ f(x)]) * \\ I(f[x \mapsto 0]) \wedge p \end{matrix}\right\} C \left\{\begin{matrix} I([x \mapsto \sigma' \circ f(x)]) * \\ I(f[x \mapsto 0]) \wedge q \end{matrix}\right\}}\ \textsc{Frame}}{\forall f.\ \left\{\begin{matrix} I([x \mapsto \sigma \circ f(x)] \circ \\ f[x \mapsto 0]) \wedge p \end{matrix}\right\} C \left\{\begin{matrix} I([x \mapsto \sigma' \circ f(x)] \circ \\ f[x \mapsto 0]) \wedge q \end{matrix}\right\}}\ \text{Lemma 1a}}{\dfrac{\forall f.\ \{I([x \mapsto \sigma] \circ f) \wedge p\}\ C\ \{I([x \mapsto \sigma'] \circ f) \wedge q\}}{I.\ \{\{[x \mapsto \sigma]\} \wedge p\}\ C\ \{\{[x \mapsto \sigma']\} \wedge q\}}\ \textsc{Enter1}}$$

A-2 Details of Examples

Turnstiles omitted in these proofs. The comments on proof trees assume they are read from the bottom up.

A-2.1 Bit Pair

Saturation lemma: $I([\_ \mapsto [i \mapsto b] \circ a]) \vdash \exists b'.\ a = [3{-}i \mapsto b']$.

Proof of bp_new**.** Let $C = (\mathsf{p} := \mathsf{alloc}\ 1;\ [\mathsf{p}] := 0)$, i.e., the body of bp_new.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\forall \phi.\ \{emp\}\ C\ \{\mathsf{p} \mapsto 0\}}\ \text{(trivial)}}{\forall \phi.\ \{emp\}\ C\ \{\{1,2\} = \{1,2\} \wedge \mathsf{p} \mapsto (0 + 2 \cdot 0)\}}\ \text{(simplify)}}{\forall \phi.\ \{emp\}\ C\ \{I([\mathsf{p}{\mapsto}[1{\mapsto}\mathit{false}, 2{\mapsto}\mathit{false}]])\}}\ \text{(definition)}}{\forall \phi.\ \{I(\phi)\}\ C\ \{I([\mathsf{p}{\mapsto}[1{\mapsto}\mathit{false}, 2{\mapsto}\mathit{false}]]) * I(\phi)\}}\ \textsc{Frame}}{\forall \phi.\ \{I(\phi)\}\ C\ \{I([\mathsf{p}{\mapsto}[1{\mapsto}\mathit{false}, 2{\mapsto}\mathit{false}]] \circ \phi)\}}\ \text{Lemma 1b}}{I.\ \{\{0\}\}\ C\ \{\{[\mathsf{p}{\mapsto}[1{\mapsto}\mathit{false}, 2{\mapsto}\mathit{false}]]\}\}}\ \textsc{Enter1}}{I.\ \{emp\}\ C\ \{B_1(\mathsf{p}, \mathit{false}) * B_2(\mathsf{p}, \mathit{false})\}}\ \text{(simplify)}$$

Proof of bp_free. Let $C = (\mathsf{free\ p})$, i.e., the body of bp_free.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\{\mathsf{p}\mapsto \_\}\ C\ \{emp\}}\ \text{(trivial)}}{\{\{1,2\}=\{1,2\}\wedge \mathsf{p}\mapsto(b_1+2\cdot b_2)\}\ C\ \{emp\}}\ \text{(simplify)}}{\{I([\mathsf{p}\mapsto[1{\mapsto}b_1,2{\mapsto}b_2]])\}\ C\ \{emp\}}\ \text{(definition)}}{\forall\phi.\ \{I([\mathsf{p}\mapsto[1{\mapsto}b_1,2{\mapsto}b_2]])*I(\phi)\}\ C\ \{I(\phi)\}}\ \textsc{Frame}}{\forall\phi.\ \{I([\mathsf{p}\mapsto[1{\mapsto}b_1,2{\mapsto}b_2]]\circ\phi)\}\ C\ \{I(\phi)\}}\ \text{Lemma 1a}}{I.\ \{\{[\mathsf{p}\mapsto[1{\mapsto}b_1,2{\mapsto}b_2]]\}\}\ C\ \{\{0\}\}}\ \textsc{Enter1}}{I.\ \{B_1(\mathsf{p},b_1)*B_2(\mathsf{p},b_2)\}\ C\ \{emp\}}\ \text{(simplify)}$$

Proof of bp_get1. Let $C = (\mathsf{x} := [\mathsf{p}])$, i.e., the body of bp_get1.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\forall b_2.\ \{\mathsf{p}\mapsto(b+2\cdot b_2)\}\ C\ \{\mathsf{p}\mapsto(b+2\cdot b_2)\wedge \mathsf{x}\%2=b\}}\ \text{(trivial)}}{\forall b_2.\ \{I([\mathsf{p}\mapsto[1{\mapsto}b,2{\mapsto}b_2]])\}\ C\ \{I([\mathsf{p}\mapsto[1{\mapsto}b,2{\mapsto}b_2]])\wedge \mathsf{x}\%2=b\}}\ \text{(definition)}}{\forall a.\ \{I([\mathsf{p}\mapsto[1{\mapsto}b]\circ a])\}\ C\ \{I([\mathsf{p}\mapsto[1{\mapsto}b]\circ a])\wedge \mathsf{x}\%2=b\}}\ \text{Saturation}}{I.\ \{\{[\mathsf{p}{\mapsto}[1{\mapsto}b]]\}\}\ C\ \{\{[\mathsf{p}{\mapsto}[1{\mapsto}b]]\}\wedge \mathsf{x}\%2=b\}}\ \text{Lemma 1c}}{I.\ \{B_1(\mathsf{p},b)\}\ C\ \{B_1(\mathsf{p},b)\wedge \mathsf{x}\%2=b\}}\ \text{(definition)}$$

Proof of bp_set1. Let $C = (\mathsf{x} := [\mathsf{p}];\ [\mathsf{p}] := \mathsf{b} + \mathsf{x}/2{*}2)$, i.e., the body of bp_set1.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\forall b_2.\ (\mathsf{p}\mapsto \mathsf{b}+(b'+2\cdot b_2)/2\cdot 2\ \vdash\ \mathsf{p}\mapsto \mathsf{b}+2\cdot b_2)}\ \text{(arithmetic)}}{\forall b_2.\ \{\mathsf{p}\mapsto \_\wedge \mathsf{x}=b'+2\cdot b_2\}\ [\mathsf{p}]:=\mathsf{b}+\mathsf{x}/2{*}2\ \{\mathsf{p}\mapsto \mathsf{b}+2\cdot b_2\}}\ \textsc{Write}}{\forall b_2.\ \{\mathsf{p}\mapsto b'+2\cdot b_2\}\ C\ \{\mathsf{p}\mapsto \mathsf{b}+2\cdot b_2\}}\ \textsc{Seq},\textsc{Read}}{\forall b_2.\ \{I([\mathsf{p}\mapsto[1{\mapsto}b',2{\mapsto}b_2]])\}\ C\ \{I([\mathsf{p}\mapsto[1{\mapsto}\mathsf{b},2{\mapsto}b_2]])\}}\ \text{(definition)}}{\forall a.\ \{I([\mathsf{p}\mapsto[1{\mapsto}b']\circ a])\}\ C\ \{I([\mathsf{p}\mapsto[1{\mapsto}\mathsf{b}]\circ a])\}}\ \text{Saturation}}{I.\ \{\{[\mathsf{p}{\mapsto}[1{\mapsto}b']]\}\}\ C\ \{\{[\mathsf{p}{\mapsto}[1{\mapsto}\mathsf{b}]]\}\}}\ \text{Lemma 1c}}{I.\ \{B_1(\mathsf{p},b')\}\ C\ \{B_1(\mathsf{p},\mathsf{b})\}}\ \text{(definition)}$$

A-2.2 Fractional Permissions

Saturation lemma: $I_{\mathrm{fp}}([\_\mapsto(v,z)\circ a])\vdash a=(v,1-z)$.

Proof of the indirect entailment. Read the following from the bottom up and consider all free variables to be universally quantified on every line.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\exists v'.\ (v,1) = (v',1) \land l \mapsto v' \dashv\vdash l \mapsto v}\ \text{(trivial)}}{I_{\mathrm{fp}}([l \mapsto (v,1)]) \dashv\vdash l \mapsto v}\ \text{(definition)}}{I_{\mathrm{fp}}([l \mapsto (v,1)]) * I_{\mathrm{fp}}(\phi_1) * \{\phi_2\} \dashv\vdash l \mapsto v * \{\phi_2\} * I_{\mathrm{fp}}(\phi_1)}\ \text{(cancellation)}}{I_{\mathrm{fp}}([l \mapsto (v,1)] \circ \phi_1) * \{\phi_2\} \dashv\vdash l \mapsto v * \{\phi_2\} * I_{\mathrm{fp}}(\phi_1)}\ \text{Lemma 1a}}{\begin{array}{c}\exists a \in (l \overset{1}{\mapsto} v).\ I_{\mathrm{fp}}(a \circ \phi_1) * \{\phi_2\} \dashv\vdash \\ \exists h \in (l \mapsto v).\ \{h \circ \phi_2\} * I_{\mathrm{fp}}(\phi_1)\end{array}}\ \text{(simplify)}}{(l \overset{1}{\mapsto} v)^{\mathsf{L}} \Rrightarrow\!\models_{I_{\mathrm{fp}} * 1} (l \mapsto v)^{\mathsf{R}}}\ \text{(definition)}$$

Proof of read.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{x \notin fv(e,e')}{\{e \mapsto e'\}\ x := [e]\ \{e \mapsto e' \land x = e'\}}\ \textsc{Read-std}}{\left\{\begin{array}{c}\exists v.\ (e',1) = (v,1) \land \\ e \mapsto v\end{array}\right\} x := [e] \left\{\begin{array}{c}\exists v.\ (e',1) = (v,1) \land \\ e \mapsto v \land x = e'\end{array}\right\}}\ \text{(simplify)}}{\{I_{\mathrm{fp}}([e \mapsto (e',1)])\}\ x := [e]\ \{I_{\mathrm{fp}}([e \mapsto (e',1)]) \land x = e'\}}\ \text{(definition)}}{\forall a.\ (\{I_{\mathrm{fp}}([e \mapsto (e',z) \circ a])\}\ x := [e]\ \{I_{\mathrm{fp}}([e \mapsto (e',z) \circ a]) \land x = e'\})}\ \text{Saturation}}{I_{\mathrm{fp}}.\ \{\{[e \mapsto (e',z)]\}\}\ x := [e]\ \{\{[e \mapsto (e',z)]\} \land x = e'\}}\ \text{Lemma 1c}}{I_{\mathrm{fp}}.\ \{e \overset{z}{\mapsto} e'\}\ x := [e]\ \{e \overset{z}{\mapsto} e' \land x = e'\}}\ \text{(definition)}$$

A-2.3 Monotonic Counter

Proof of mc_new**.** Let $C = (\mathsf{c} := \mathsf{alloc}\ 1;\ [\mathsf{c}] := 0)$, i.e., the body of mc_new.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\{emp\}\ C\ \{\mathsf{c} \mapsto 0\}}\ \text{(trivial)}}{\{emp\}\ C\ \{\exists j \geq 0.\ \mathsf{c} \mapsto j\}}\ \text{(instantiate)}}{\{emp\}\ C\ \{I([\mathsf{c} \mapsto 0])\}}\ \text{(definition)}}{\forall \phi.\ \{I(\phi)\}\ C\ \{I([\mathsf{c} \mapsto 0]) * I(\phi)\}}\ \textsc{Frame}}{\forall \phi.\ \{I(\phi)\}\ C\ \{I([\mathsf{c} \mapsto 0] \circ \phi)\}}\ \text{Lemma 1b}}{I.\ \{\{0\}\}\ C\ \{\{[\mathsf{c} \mapsto 0]\}\}}\ \textsc{Enter1}}{I.\ \{emp\}\ C\ \{MC(\mathsf{c}, 0)\}}\ \text{(definition)}$$

Proof of mc_read. Let $C = (\mathsf{x} := [\mathsf{c}])$, i.e., the body of mc_read.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\forall i'.\, \forall j \geq max(i,i').\, (\mathsf{x} = j \vdash \mathsf{x} \geq i)}\,(\text{arithmetic})}{\forall i'.\, \forall j \geq max(i,i').\, \{\mathsf{c} \mapsto j\}\ C\ \{\mathsf{c} \mapsto j \wedge \mathsf{x} \geq i\}}\,\textsc{Read}}{\forall i'.\, \forall j \geq max(i,i').\, \{\mathsf{c} \mapsto j\}\ C\ \{\exists j \geq max(i,i').\, \mathsf{c} \mapsto j \wedge \mathsf{x} \geq i\}}\,(\text{instantiate})}{\forall i'.\, \{\exists j \geq max(i,i').\, \mathsf{c} \mapsto j\}\ C\ \{\exists j \geq max(i,i').\, \mathsf{c} \mapsto j \wedge \mathsf{x} \geq i\}}\,\textsc{Exists}}{I.\, \{\{[\mathsf{c} \mapsto i]\}\}\ C\ \{\{[\mathsf{c} \mapsto i]\} \wedge \mathsf{x} \geq i\}}\,\text{Lemma 1c}}{I.\, \{MC(\mathsf{c}, i)\}\ C\ \{MC(\mathsf{c}, i) \wedge \mathsf{x} \geq i\}}\,(\text{definition})$$

Proof of mc_inc. Let $C = (\mathsf{x} := [\mathsf{c}];\ [\mathsf{c}] := \mathsf{x}{+}1)$, i.e., the body of mc_inc.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\forall j.\, \{\mathsf{c} \mapsto j\}\ C\ \{\mathsf{c} \mapsto j+1\}}\,(\text{trivial})}{\forall i'.\, \forall j \geq max(i,i').\, \{\mathsf{c} \mapsto j\}\ C\ \{\mathsf{c} \mapsto j+1\}}\,(\text{weakening})}{\forall i'.\, \forall j \geq max(i,i').\, \{\mathsf{c} \mapsto j\}\ C\ \{\exists j \geq max(i+1,i').\, \mathsf{c} \mapsto j\}}\,(\text{instantiate})}{\forall i'.\, \{\exists j \geq max(i,i').\, \mathsf{c} \mapsto j\}\ C\ \{\exists j \geq max(i+1,i').\, \mathsf{c} \mapsto j\}}\,\textsc{Exists}}{I.\, \{\{[\mathsf{c} \mapsto i]\}\}\ C\ \{\{[\mathsf{c} \mapsto i+1]\}\}}\,\text{Lemma 1c}}{I.\, \{MC(\mathsf{c}, i)\}\ C\ \{MC(\mathsf{c}, i+1)\}}\,(\text{definition})$$

where $(\star)$ is a proof of the arithmetic fact that

$$j \geq max(i,i') \vdash j+1 \geq max(i+1,i')$$

A-2.4 Abstract Fractional Permissions

Proof of clone(c).

$$\dfrac{\dfrac{\dfrac{\dfrac{I.\, \{Coll(\mathsf{c}, V)\}\ \ \{Coll(\mathsf{c}, V) * Coll(\mathsf{ret}, V)\}}{I.\, \{I'([\mathsf{c} \mapsto (V,1)])\}\ \ \{I'([\mathsf{c} \mapsto (V,1)]) * Coll(\mathsf{ret}, V)\}}\,(\text{definition})}{\forall \phi.\, I.\, \{I'([\mathsf{c} \mapsto (V,z)] \circ \phi)\}\ \ \{I'([\mathsf{c} \mapsto (V,z)] \circ \phi) * Coll(\mathsf{ret}, V)\}}\,\text{Saturation etc.}}{I' \succ I.\, \{\{[\mathsf{c} \mapsto (V,z)]\}^{\mathsf{L}}\}\ \ \{\{[\mathsf{c} \mapsto (V,z)]\} \times Coll(\mathsf{ret}, V)\}}\,\textsc{Enter1Stack}}{I' \succ I.\, \{FColl^z(\mathsf{c}, V)^{\mathsf{L}}\}\ \ \{FColl^z(\mathsf{c}, V)^{\mathsf{L}} * FColl^1(\mathsf{ret}, V)^{\mathsf{L}}\}}\,\textsc{ROC-Indirect}$$

A-3 Conjunction Rule and Friends

The conjunction rule is most often written as

$$\dfrac{S \vdash I.\, \{P_1\}\ C\ \{Q_1\} \qquad S \vdash I.\, \{P_2\}\ C\ \{Q_2\}}{S \vdash I.\, \{P_1 \wedge P_2\}\ C\ \{Q_1 \wedge Q_2\}}$$

Despite not being used much in separation logic, the rule has received a lot of attention in the literature, e.g., [O'H07, DYGW11, GBC11], perhaps because the circumstances under which it holds are theoretically interesting. We shall here investigate restrictions on interpretations I that are sufficient for the conjunction rule or related rules to hold in fictional separation logic.

We will work with the following reformulation of the conjunction rule, which is easily shown equivalent to the previous one

$$\frac{S \vdash I.\ \{P\}\ C\ \{Q_1\} \qquad S \vdash I.\ \{P\}\ C\ \{Q_2\}}{S \vdash I.\ \{P\}\ C\ \{Q_1 \wedge Q_2\}}\ \textsc{Conjunction}$$

Recall the definition of I^ϕ (Definition A-1 on page 2).

Lemma A-1. *If, for all ϕ, I^ϕ distributes over conjunctions, then the conjunction rule holds.*

Proof.

$$\dfrac{\dfrac{\dfrac{I.\ \{P\}\ C\ \{Q_1\}}{\forall\phi.\ \{I^\phi(P)\}\ C\ \{I^\phi(Q_1)\}} \quad \dfrac{I.\ \{P\}\ C\ \{Q_2\}}{\forall\phi.\ \{I^\phi(P)\}\ C\ \{I^\phi(Q_2)\}}\ \text{(definition)}}{\forall\phi.\ \{I^\phi(P)\}\ C\ \{I^\phi(Q_1) \wedge I^\phi(Q_2)\}}\ \textsc{Conjunction-std}}{\dfrac{\forall\phi.\ \{I^\phi(P)\}\ C\ \{I^\phi(Q_1 \wedge Q_2)\}}{I.\ \{P\}\ C\ \{Q_1 \wedge Q_2\}}\ \text{(definition)}}\ \text{(assumption)}$$

The Conjunction-std rule applied above is the one for standard triples, which is known to hold. □

Note that $I^\phi(Q_1 \wedge Q_2) \vdash I^\phi(Q_1) \wedge I^\phi(Q_2)$ always holds; whereas the converse direction does not always hold. With the following definitions, we can give necessary and sufficient conditions for when it holds.

Definition A-1. I is *relationally frame-injective* if when $h \in I(\sigma \circ \phi)$ and $h \in I(\sigma' \circ \phi)$ then $\sigma = \sigma'$. ◇

Lemma A-2. *I is relationally frame-injective iff, for all ϕ, I^ϕ distributes over conjunctions.*

Proof. From left to right, assume $h \in I^\phi(P)$ and $h \in I^\phi(Q)$. The goal is to prove $h \in I^\phi(P \wedge Q)$. From our assumptions, we get the existence of $\sigma \in P$ and $\sigma' \in Q$ such that $h \in I(\sigma \circ \phi)$ and $h \in I(\sigma' \circ \phi)$. From the last two facts, relational frame-injectivity implies $\sigma = \sigma'$, so we can use this value as our witness to prove the goal.

From right to left, assume $h \in I(\sigma \circ \phi)$ and $h \in I(\sigma' \circ \phi)$. The goal is to prove $\sigma = \sigma'$. Instantiate the distributivity lemma as $I^\phi(\{\sigma\}) \wedge I^\phi(\{\sigma'\}) \vdash I^\phi(\{\sigma\} \wedge \{\sigma'\})$ and specialize this to h. The right-hand side of that entailment implies that $\sigma = \sigma'$, and the left-hand side is exactly our assumption. □

Definition A-2.

- I is *relationally injective* if when $h \in I(\sigma)$ and $h \in I(\sigma')$ then $\sigma = \sigma'$.
- Σ is *cancellative* if when $\sigma' \doteq \sigma \circ \sigma_1$ and $\sigma' \doteq \sigma \circ \sigma_2$ then $\sigma_1 = \sigma_2$. ◇

Lemma A-3. *For $I : \Sigma \setminus \Sigma'$, if Σ is cancellative and I is relationally injective, then I is relationally frame-injective.*

A-3.1 The Recombination Rule

Among our examples, the conjunction rule holds for bit pair and fractional permissions but not for monotonic counters. We can generalize the rule, however. If for some connective $(\square)$ we have $I^\phi(Q_1) \wedge I^\phi(Q_2) \vdash I^\phi(Q_1 \,\square\, Q_2)$, for all ϕ, then the following holds

$$\frac{S \vdash I.\ \{P\}\ C\ \{Q_1\} \qquad S \vdash I.\ \{P\}\ C\ \{Q_2\}}{S \vdash I.\ \{P\}\ C\ \{Q_1 \,\square\, Q_2\}}\ \textsc{Recombination}$$

Proof. Analogous to the proof of Lemma A-1. □

Lemma A-4. *For the monotonic counters, the* Recombination *rule holds for* $(\square) = (*)$.

Proof. Recall the algebra and interpretation for monotonic counters:

$$\Sigma = \mathit{loc} \xrightarrow{\text{fin}} \mathbb{Z}_\perp \text{ where composition in } \mathbb{Z} \text{ is } \mathit{max}$$
$$I(f) = \forall_*\, c \in \mathit{supp}(f).\ \exists j \geq f(c).\ c \mapsto j$$

Notice a fact about I: If $h \in I(\sigma)$ and $h \in I(\sigma')$ then $h \in I(\sigma \circ \sigma')$.

To prove the prerequisite of Recombination, assume $h \in I^\phi(Q_1)$ and $h \in I^\phi(Q_2)$. The goal is to prove $h \in I^\phi(Q_1 * Q_2)$. From the two assumptions, we get the existence of $\sigma_1 \in Q_1$ and $\sigma_2 \in Q_2$ such that $h \in I(\sigma_1 \circ \phi)$ and $h \in I(\sigma_2 \circ \phi)$. To prove our goal, we pick the witness $(\sigma_1 \circ \sigma_2) \in (Q_1 * Q_2)$ and see that $h \in I((\sigma_1 \circ \phi) \circ (\sigma_2 \circ \phi)) = I(\sigma_1 \circ \sigma_2 \circ \phi)$ thanks to the fact about I and idempotence of $(\circ)$. □

A-3.2 Indirect Entailment Rules

Figure A-1 summarizes the usual BI logic rules and whether they are sound for indirect entailment. As we can see, some rules are missing. In particular, there is no $\wedge$-introduction rule. But such a rule is a direct consequence of the conjunction rule when I interprets into *heap*:

Lemma A-5. *If the conjunction rule holds for I or if I^ϕ distributes over conjunctions for all ϕ, then $\wedge$-introduction holds in the logic of $(\models_I)$.*

Proof. To derive it from the conjunction rule, choose $C = \mathit{skip}$ and see that the definition of an indirect triple collapses to that of an indirect entailment. The derivation from distributivity is analogous to the proof of Lemma A-1.□

Interestingly, the converses of this lemma do not hold. Before discussing a counterexample, let us consider an even stronger condition on I: the situation where $(\models_I) = (\vdash)$. This equality of relations is equivalent to having for all P and Q,

$$\frac{P \vdash Q}{P \models_I Q} \qquad \text{and} \qquad \frac{P \models_I Q}{P \vdash Q}$$

$$\frac{P \models_I Q \quad Q \models_I R}{P \models_I R} \qquad \frac{P \vdash Q}{P \models_I Q} \qquad \frac{\vdash I.\ \{P\}\ \mathit{skip}\ \{Q\}}{P \models_I Q}$$

$$\frac{\forall x.\ (P(x) \models_I Q)}{\exists x.\ P(x) \models_I Q} \quad \frac{P \wedge Q \models_I R}{P \models_I Q \Rightarrow R} \quad \frac{P \models_I Q \mathrel{-\!\!*} R}{P * Q \models_I R} \quad \frac{P \models_I P' \qquad Q \models_I Q'}{P * Q \models_I P' * Q'}$$

The following are derivable because standard entailment implies indirect entailment.

$$\frac{}{P \models_I P} \qquad \frac{}{P \models_I \top} \qquad \frac{}{\bot \models_I P} \qquad \frac{P \models_I Q \wedge R}{P \models_I Q} \qquad \frac{P \models_I \forall x.\ Q(x)}{P \models_I Q(y)}$$

$$\frac{}{P \models\!\!\mid_I P * \mathit{emp}} \qquad \frac{}{P * Q \models_I Q * P} \qquad \frac{}{(P * Q) * R \models_I P * (Q * R)}$$

The following are derivable because disjunction is just a special case of $\exists$.

$$\frac{P \models_I Q}{P \models_I Q \vee R} \qquad \frac{P \models_I R \qquad Q \models_I R}{P \vee Q \models_I R}$$

Rules from BI logic that are unsound here:

$$\cancel{\frac{P \models_I Q \qquad P \models_I R}{P \models_I Q \wedge R}} \qquad \cancel{\frac{\forall x.\ (P \models_I Q(x))}{P \models_I \forall x.\ Q(x)}} \qquad \cancel{\frac{P \models_I Q \quad P \models_I Q \Rightarrow R}{P \models_I R}} \qquad \cancel{\frac{P * Q \models_I R}{P \models_I Q \mathrel{-\!\!*} R}}$$

Figure A-1: Inference rules for indirect entailment. Symmetric ones have been elided.

The former of these rules always holds, while the latter appears to be connected with the following notion.

Definition A-3. I is *completable* if for any σ there exists ϕ and h such that $h \in I(\sigma \circ \phi)$. $\diamond$

This is a fairly weak property, satisfied by the bit pair, fractional permissions and monotonic counter examples.

Lemma A-6. $(\models_I) = (\vdash)$ *if and only if* I *is completable and* $\wedge$*-introduction holds for* $(\models_I)$.

Proof. Assume first that the relations are equal. Completability follows from the special case that $\{\sigma\} \models_I \bot$ implies $\{\sigma\} \vdash \bot$. It is trivial that $\wedge$-introduction holds.

For the other direction of the lemma, assume $P \models_I Q$ and note that this is equivalent to $\forall \sigma \in P.\ (\{\sigma\} \models_I Q)$ by the existential rule from Figure A-1. We must show that if $\sigma \in P$ then $\sigma \in Q$. The premise and our assumption allow us to conclude that $\{\sigma\} \models_I Q$. Now instantiate the $\wedge$-introduction rule with the valid premises $\{\sigma\} \models_I \{\sigma\}$ and $\{\sigma\} \models_I Q$ to conclude that $\{\sigma\} \models_I \{\sigma\} \wedge Q$. Specialize this proposition with ϕ and h obtained by completing σ so it says that if $h \in I(\sigma \circ \phi)$ then $h \in I^{\phi}(\{\sigma\} \wedge Q)$. The premise of that is true by the construction of ϕ and h, and the conclusion implies that $\sigma \in Q$. $\square$

Even though $(\models_I) = (\vdash)$ seems like a very strong condition on I, it is not enough to guarantee that the conjunction rule holds, as demonstrated by the following counterexample.

Example A-1. Take the separation algebra $\Sigma = \{1, 2\}_{\emptyset\bot}$. Define $I : \Sigma \searrow$ *heap* by

$$\begin{aligned} I(0) &= \mathit{emp} \\ I(1) &= \{[0 \mapsto 0], [0 \mapsto 1]\} \\ I(2) &= \{[0 \mapsto 0], [0 \mapsto 2]\} \end{aligned}$$

The mapping of 0 serves only to satisfy the side condition on $(\searrow)$ and plays no interesting role here. The other mappings are chosen such that they are different but share a common element.

This common element means that I is not relationally injective. It also makes the conjunction rule fail since we have

$$\begin{aligned} &\vdash I.\ \{\{1\}\}\ [0] := 0\ \{\{1\}\} \text{ and} \\ &\vdash I.\ \{\{1\}\}\ [0] := 0\ \{\{2\}\} \text{ but not} \\ &\vdash I.\ \{\{1\}\}\ [0] := 0\ \{\{1\} \wedge \{2\}\} \end{aligned}$$

since the last postcondition implies false.

But this I *does* satisfy $(\models_I) = (\vdash)$. To show this, we must assume $P \models_I Q$ and prove $P \vdash Q$, i.e., $P \subseteq Q$. There is a finite number of cases to check, and we can reduce their number somewhat. The conclusion is automatically true if $P = \emptyset$ or $Q = \{0, 1, 2\}$ among several other cases. It is only when subset inclusion does not hold that we must show that indirect entailment also fails. Observe that for any I,

$$\frac{P \subseteq P' \qquad P \not\models_I Q \qquad Q' \subseteq Q}{P' \not\models_I Q'}$$

This means that if we can disprove the indirect entailment for Q being $\{0, 1\}$, $\{0, 2\}$ or $\{1, 2\}$, then we have disproved it for all values of Q except the one we are not interested in. Knowing Q to be, for example, $\{0, 1\}$, we only have to check the indirect entailment for $P = \{2\}$ since any $P' \not\subseteq Q$ will a superset of this value. So there are just three cases to prove.

- $\{0\} \not\models_I \{1, 2\}$. If the frame is 0, there is only one choice of heap on the left side, and it cannot be recovered on the right. If the frame is 1 or 2, it will not compose with any of the two choices on the right.
- $\{1\} \not\models_I \{0, 2\}$. The frame must be 0 to compose with 1. The heap is constrained on the left to be $[0 \mapsto 0]$ or $[0 \mapsto 1]$, but the heap $[0 \mapsto 1]$ does not exist in neither $I(0)$ nor $I(2)$.
- Symmetric to the previous case. ◇

A-4 Further Examples

A-4.1 Concrete Client of Two Modules

To show in more detail how a client can work with separating products, we will here discuss a concrete piece of client code. It is the same example as in Section 4.1 but with some code inserted to make it concrete.

Let I_{bp} and I_{mc} be the interpretations functions for bit pairs and monotonic counters respectively. Then consider the following function:

```
f(bp, ctr) {
  b := call bp_get1(bp);
  if b = 0 then
    call mc_inc(ctr);
  call bp_set1(bp, 1)
}
```

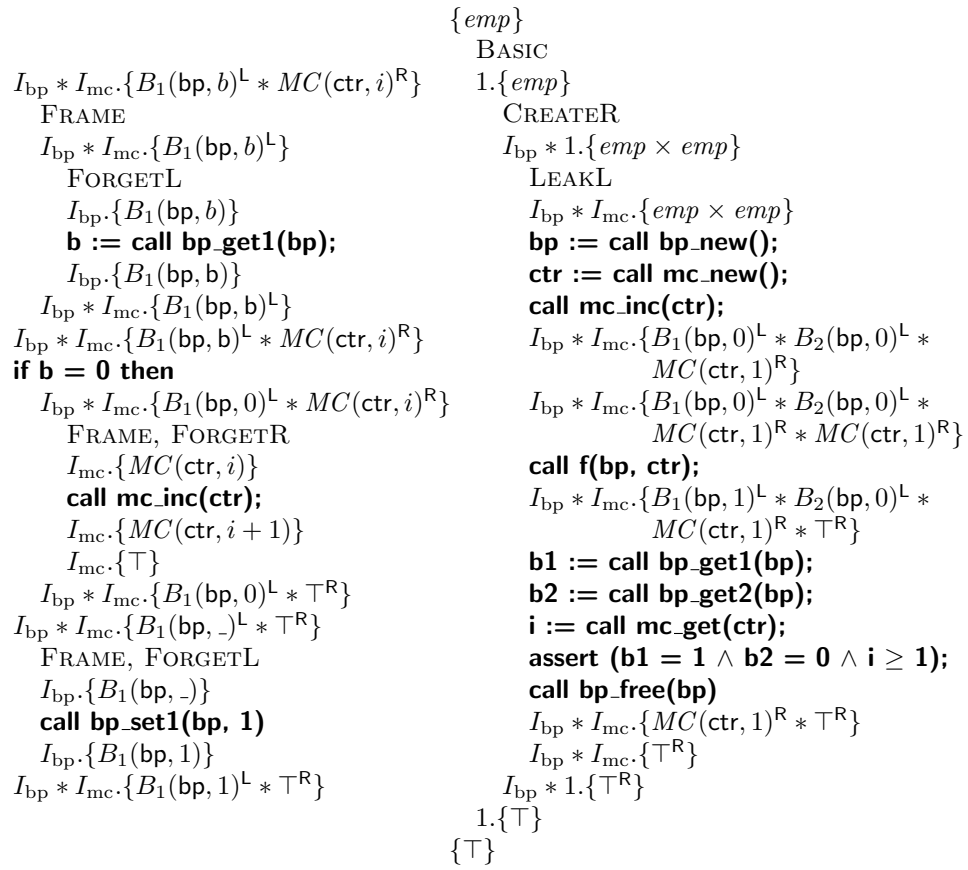

Figure A-2: Proof sketch for client code example

We can specify it as

$$I_{\mathrm{bp}} * I_{\mathrm{mc}}.\ \{B_1(\mathsf{bp}, b)^{\mathsf{L}} * MC(\mathsf{ctr}, i)^{\mathsf{R}}\}\ \mathsf{f}(\mathsf{bp}, \mathsf{ctr})\ \{B_1(\mathsf{bp}, 1)^{\mathsf{L}} * \top^{\mathsf{R}}\},$$

which is a fairly weak specification. Its verification is shown in the first column of Figure A-2.

A caller of this function might look as follows.

```
bp := call bp_new();
ctr := call mc_new();
call mc_inc(ctr);
call f(bp, ctr);
b1 := call bp_get1(bp);
b2 := call bp_get2(bp);
i := call mc_get(ctr);
assert (b1 = 1 and b2 = 0 and i >= 1);
call bp_free(bp)
```

If the above code is called C, then the second column in Figure A-2 shows a verification of $\{emp\}\ C\ \{\top\}$. The steps of using FRAME and FORGET around each function call as we saw in the left column are now elided just as use of FRAME is traditionally left implicit in separation logic proof narrations. The conclusion here is that after the initial set-up of separating products to deal with multiple modules, verification with fictional separation logic is very similar to standard separation logic.

A-4.2 Weak-update type system

Most work on separation logic ignores the type system of the underlying programming language, if it has any. A notable exception is the work of Tan et al. [TSFC09], which presents a dialect of separation logic that mixes types and assertions. We will see in this section how to reconstruct their logic on top of fictional separation logic, saving a lot of effort compared to proving soundness directly with respect to the operational semantics of the language.

Types can be easier to work with than assertions in many cases where only memory safety is to be verified. This comes up, e.g., when using the foreign-function interface of a high-level language.

Recall that programming language values *val* is the disjoint union of integers, Booleans and locations. Assume the injections to *val* from Booleans and integers respectively are named val_{int} and val_{bool}. Define types as follows:

$$\tau : \mathit{type} ::= \mathsf{int} \mid \mathsf{bool} \mid \mathsf{ref}\ \tau.$$

Theorem A-1. *There exists a separation algebra Σ, a predicate $\langle v : \tau \rangle : \mathcal{P}(\Sigma)$, and an interpretation $I : \Sigma \setminus \mathit{heap}$ such that the rules in Figure A-3 are valid.*

Proof. Choose the existentials as follows.

$$\begin{aligned}
\Sigma &= \mathit{loc} \xrightarrow{\text{fin}} \mathit{type}_{=\bot} \\
\langle v : \tau \rangle &= \top * \mathsf{case}\ v, \tau\ \mathsf{of} \\
&\qquad \mid \mathit{val}_{\text{int}}(n), \mathsf{int} \Rightarrow \mathit{emp} \\
&\qquad \mid \mathit{val}_{\text{bool}}(b), \mathsf{bool} \Rightarrow \mathit{emp} \\
&\qquad \mid \mathit{val}_{\text{loc}}(l), \mathsf{ref}\ \tau \Rightarrow \{[l \mapsto \tau]\} \\
&\qquad \mid _, _ \Rightarrow \bot \\
I(\Psi) &= \forall_* l \in \mathit{supp}(\Psi).\ \exists v.\ l \mapsto v \wedge \Psi \in \langle v : \Psi(l) \rangle
\end{aligned}$$

Details of the proofs are in Section A-4.2.1. □

Figure A-3 contains the essential ingredients needed in a weak-update type system. Proving the rules is not trivial, but it is a very minimal theory,

$$\frac{}{\langle v : \mathsf{int}\rangle \dashv\vdash \exists n.\ v = val_{\mathrm{int}}(n)} \qquad \frac{}{\langle v : \mathsf{bool}\rangle \dashv\vdash \exists b.\ v = val_{\mathrm{bool}}(b)}$$

$$\frac{}{\langle e_1 : \mathsf{int}\rangle \wedge \langle e_2 : \mathsf{int}\rangle \vdash \langle e_1 + e_2 : \mathsf{int}\rangle} \quad \cdots \quad \frac{}{\langle e_1 : \mathsf{int}\rangle \wedge \langle e_2 : \mathsf{int}\rangle \vdash \langle e_1 \geq e_2 : \mathsf{bool}\rangle}$$

$$\frac{}{\vdash I.\ \{\langle e : \mathsf{ref}\ \tau\rangle\}\ x := [e]\ \{\langle x : \tau\rangle\}}\ \text{W-Read}$$

$$\frac{}{\vdash I.\ \{\langle e : \mathsf{ref}\ \tau\rangle * \langle e' : \tau\rangle\}\ [e] := e'\ \{\top\}}\ \text{W-Write}$$

$$\frac{}{\langle v : \tau\rangle \times l \mapsto v \models_{I*1} \langle l : \mathsf{ref}\ \tau\rangle \times emp}\ \text{W-s2w}$$

$$\frac{}{\langle v : \tau\rangle \dashv\vdash \langle v : \tau\rangle * \top} \qquad \frac{P : \mathcal{P}(\Sigma)}{P \vdash P * P}\ \text{W-Dupl.}$$

Figure A-3: The essentials of a type system for weak updates

free of distractions. Although it does not yet look like the system of Tan et al., or any other type system, we can derive such a presentation using standard building blocks from fictional separation logic.

We first define stack type environments, which conveniently can be modelled as separation algebras although this is not essential for the theory.

$$\begin{aligned} \Gamma : env &\triangleq var \overset{\text{fin}}{\Rightarrow} type_{=\bot} \\ [\![\Gamma]\!] &\triangleq \top * \forall_* x \in supp(\Gamma).\ \langle x : \Gamma(x)\rangle \\ e :_\Gamma \tau &\triangleq [\![\Gamma]\!] \vdash \langle e : \tau\rangle. \end{aligned}$$

As usual, we implicitly lift $\langle v : \tau\rangle : \mathcal{P}(\Sigma)$ to $\langle e : \tau\rangle : asn(\Sigma)$.

Theorem A-2. *The rules in Figure A-4 are valid.*

Proof. The only non-trivial rule is W-Hom-$\circ$. It is proved in Section A-4.2.2. □

We can now define a triple that combines the type system and standard separation logic:

$$\{\Gamma, P\}\ C\ \{\Gamma', P'\} \triangleq I * 1.\ \{[\![\Gamma]\!] \times P\}\ C\ \{[\![\Gamma']\!] \times P'\}.$$

This hybrid triple satisfies the rules in Figure A-5. The rules are similar but not identical to those of Tan et al. [TSFC09] – we have made a few

$$\frac{e_1 :_\Gamma \mathsf{int} \qquad e_2 :_\Gamma \mathsf{int}}{(e_1 + e_2) :_\Gamma \mathsf{int}} \qquad \cdots \qquad \frac{e_1 :_\Gamma \mathsf{int} \qquad e_2 :_\Gamma \mathsf{int}}{(e_1 \geq e_2) :_\Gamma \mathsf{bool}}$$

$$\frac{}{[\![\Gamma \circ \Gamma']\!] \vdash [\![\Gamma]\!]}\ \text{W-WEAKEN} \qquad \frac{}{[\![[x \mapsto \tau]]\!] \dashv\vdash \langle x : \tau \rangle}\ \text{W-W1}$$

$$\frac{}{[\![0]\!] \dashv\vdash \top}\ \text{W-HOM-0} \qquad \frac{}{[\![\Gamma \circ \Gamma']\!] \dashv\vdash [\![\Gamma]\!] * [\![\Gamma']\!]}\ \text{W-HOM-}\circ$$

Figure A-4: Rules for stack type environments

simplifications. The strong-heap rules are missing some side conditions that seem unnecessary because we do not require all variables to be mentioned in Γ. There is no Ψ-context on the rules because it is not necessary.

Theorem A-3. *The rules in Figure A-5 are valid.*

Proof. The proofs are easy because all the ingredients are already provided by the theory of separating products. The W-*' rules are proved from the rules in Figure A-3 by framing on the stack type environment, then extending them to the separating product with ForgetL. Similarly, the S-*' rules are proved from the standard separation logic rules by applying ForgetR, then framing on a stack environment. The structural and control-flow rules are just special cases of the standard rules we get automatically from the fictional separation logic framework. Details can be found in Section A-4.2.3.□

The purpose of this example was to show how much work can be saved by not building a theory from scratch but assembling most of it from the highly composable theories of separation algebras and separating products. This separates the essential core of the theory, Figure A-3, from the complete system. It also allows us to assemble the theories differently. Instead of combining the type system with a standard separation logic, we could combine it with a separation logic for fractional permissions or any other interpretation from this text.

A-4.2.1 Detailed proofs of Figure A-3.

Proof (of W-Dupl.*).* Follows from idempotence of composition in Σ. □

Lemma A-1. *For any $f : A \overset{\text{fin}}{\Rightarrow} \Sigma$, if $f(a) = \sigma$ then $f = f[a \mapsto 0] \circ [a \mapsto \sigma]$.*

Definition A-1. $I_{\Psi'}(\Psi) \triangleq \forall_* l \in \mathit{supp}(\Psi).\ \exists v.\ l \mapsto v \wedge \Psi' \in \langle v : \Psi(l) \rangle$. ◇

$$\frac{x \notin fv(e, e')}{\vdash \{\Gamma, e \mapsto e'\}\ x := [e]\ \{\Gamma[x{\mapsto}0], e \mapsto e' \wedge x = e'\}}\ \text{S-Read'}$$

$$\frac{}{\vdash \{\Gamma, e \mapsto \_\}\ [e] := e'\ \{\Gamma, e \mapsto e'\}}\ \text{S-Write'}$$

$$\frac{}{\vdash \{\Gamma, emp\}\ x := \mathsf{alloc}\ 1\ \{\Gamma[x{\mapsto}0], x \mapsto \_\}}\ \text{S-Alloc'}$$

$$\frac{}{\vdash \{\Gamma, e \mapsto \_\}\ \mathsf{free}\ e\ \{\Gamma, emp\}}\ \text{S-Free'}$$

$$\frac{}{\vdash \{\Gamma, Q[e/x]\}\ x := e\ \{\Gamma[x{\mapsto}0], Q\}}\ \text{S-Assign'}$$

$$\frac{\mathit{modifies}(C)\ \#\ fv(R) \qquad S \vdash \{\Gamma, P\}\ C\ \{\Gamma, Q\}}{S \vdash \{\Gamma, P * R\}\ C\ \{\Gamma, Q * R\}}\ \text{Frame'}$$

$$\frac{S \vdash \{\Gamma, P \wedge e\}\ C\ \{\Gamma, P\}}{S \vdash \{\Gamma, P\}\ \mathsf{while}\ e\ \mathsf{do}\ C\ \{\Gamma, P \wedge \neg e\}}\ \text{While'}$$

$$\frac{e :_\Gamma \tau}{\{\Gamma, x \mapsto e\}\ \mathit{skip}\ \{\Gamma[x \mapsto \mathsf{ref}\ \tau], emp\}}\ \text{s2w'}$$

$$\frac{e :_\Gamma \mathsf{ref}\ \tau}{\vdash \{\Gamma, emp\}\ x := [e]\ \{\Gamma[x \mapsto \tau], emp\}}\ \text{W-Read'}$$

$$\frac{e :_\Gamma \mathsf{ref}\ \tau \qquad e' :_\Gamma \tau}{\vdash \{\Gamma, emp\}\ [e] := e'\ \{\Gamma, emp\}}\ \text{W-Write'}$$

$$\frac{e :_\Gamma \tau}{\vdash \{\Gamma, emp\}\ x := \mathsf{alloc}\ 1; [x] := e\ \{\Gamma[x \mapsto \mathsf{ref}\ \tau], emp\}}\ \text{W-Alloc'}$$

$$\frac{e :_\Gamma \tau}{\vdash \{\Gamma, emp\}\ x := e\ \{\Gamma[x \mapsto \tau], emp\}}\ \text{W-Assign'}$$

Figure A-5: An adaptation of the inference rules by Tan et al. [TSFC09]. Rules for sequence, conditional, existential and consequence are not shown.

So we have $I(\Psi) \dashv\vdash I_\Psi(\Psi)$, and for any Ψ', the interpretation $I_{\Psi'}$ satisfies the condition for using Lemma 1.

Lemma A-2. *If* $\Psi \in \langle l : \mathsf{ref}\ \tau \rangle$*, then*

$$I(\Psi \circ \phi) \dashv\vdash (\exists v.\, l \mapsto v \wedge \Psi \circ \phi \in \langle v : \tau \rangle) * I_{\Psi\circ\phi}((\Psi \circ \phi)[l \mapsto 0]).$$

Proof. The assumption that $\Psi \in \langle l : \mathsf{ref}\ \tau \rangle$ implies that $(\Psi \circ \phi)(l) = \tau$, so Lemma A-1 is applicable. (We always know that $\Psi \circ \phi$ is defined since this fact is implied by both sides of the bi-entailment.) Now we can prove the lemma:

$$\begin{aligned}
I(\Psi \circ \phi) &\dashv\vdash && \text{(definition)} \\
I_{\Psi\circ\phi}(\Psi \circ \phi) &\dashv\vdash && \text{Lemma A-1} \\
I_{\Psi\circ\phi}([l \mapsto \tau] \circ (\Psi \circ \phi)[l \mapsto 0]) &\dashv\vdash && \text{Lemma 1a} \\
I_{\Psi\circ\phi}([l \mapsto \tau]) * I_{\Psi\circ\phi}((\Psi \circ \phi)[l \mapsto 0]) &\dashv\vdash && \text{(definition)} \\
(\exists v.\, l \mapsto v \wedge \Psi \circ \phi \in \langle v : \tau \rangle) * I_{\Psi\circ\phi}((\Psi \circ \phi)[l \mapsto 0]) & && \square
\end{aligned}$$

Proof (of W-Write*).*

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \{e \mapsto \_\}\ [e] := e'\ \{e \mapsto e'\}}\ \textsc{Write-std}}{\vdash \forall \phi, \Psi.\ \left\{\begin{array}{l}\Psi \in (\langle e : \mathsf{ref}\ \tau \rangle * \langle e' : \tau \rangle) \wedge \\ e \mapsto \_ * I_{\Psi\circ\phi}((\Psi\circ\phi)[e \mapsto 0])\end{array}\right\}\ [e] := e'\ \left\{\begin{array}{l}\Psi \in (\langle e : \mathsf{ref}\ \tau \rangle * \langle e' : \tau \rangle) \wedge \\ e \mapsto e' * I_{\Psi\circ\phi}((\Psi\circ\phi)[e \mapsto 0])\end{array}\right\}}\ \textsc{Frame}}{\vdash \forall \phi, \Psi.\ \{\Psi \in (\langle e : \mathsf{ref}\ \tau \rangle * \langle e' : \tau \rangle) \wedge I(\Psi \circ \phi)\}\ [e] := e'\ \{I(\Psi \circ \phi)\}}\ \text{Lemma A-2}}{\vdash \forall \phi.\ \{\exists \Psi \in (\langle e : \mathsf{ref}\ \tau \rangle * \langle e' : \tau \rangle).\ I(\Psi \circ \phi)\}\ [e] := e'\ \{\exists \Psi.\ I(\Psi \circ \phi)\}}\ \textsc{Exists}}{\vdash I.\ \{\langle e : \mathsf{ref}\ \tau \rangle * \langle e' : \tau \rangle\}\ [e] := e'\ \{\top\}}\ \text{(definition)}$$

$\square$

Proof (of W-Read*).*

$$\dfrac{\dfrac{\dfrac{}{\vdash \forall \phi, \Psi, v.\ \left\{\begin{array}{l}\Psi \in \langle e : \mathsf{ref}\ \tau \rangle \wedge e \mapsto v \wedge \\ \Phi \circ \phi \in \langle v : \tau \rangle * \\ I_{\Psi\circ\phi}((\Psi\circ\phi)[e \mapsto 0])\end{array}\right\}\ x := [e]\ \left\{\begin{array}{l}x = v \wedge \exists l. \\ \Psi \in \langle l : \mathsf{ref}\ \tau \rangle \wedge l \mapsto v \wedge \\ \Phi \circ \phi \in \langle v : \tau \rangle * \\ I_{\Psi\circ\phi}((\Psi\circ\phi)[l \mapsto 0])\end{array}\right\}}\ \textsc{Read-std}}{\vdash \forall \phi. \forall \Psi.\ \{\Psi \in \langle e : \mathsf{ref}\ \tau \rangle \wedge I(\Psi \circ \phi)\}\ x := [e]\ \{\Psi \circ \phi \in \langle x : \tau \rangle \wedge I(\Psi \circ \phi)\}}\ \text{Lemma A-2}}{\vdash \forall \phi. \forall \Psi.\ \{\Psi \in \langle e : \mathsf{ref}\ \tau \rangle \wedge I(\Psi \circ \phi)\}\ x := [e]\ \{I^\phi \langle x : \tau \rangle\}}\ \text{(instantiate)}}{\vdash \forall \phi.\ \{\exists \Psi \in \langle e : \mathsf{ref}\ \tau \rangle.\ I(\Psi \circ \phi)\}\ x := [e]\ \{I^\phi \langle x : \tau \rangle\}}\ \textsc{Exists}}{\vdash I.\ \{\langle e : \mathsf{ref}\ \tau \rangle\}\ x := [e]\ \{\langle x : \tau \rangle\}}\ \text{(definition)}$$

Notice that the existential in the postcondition is instantiated not to Ψ but to $\Psi \circ \phi$. $\square$

Proof (of W-S2W*).*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\forall\phi,\Psi.\ (\Psi[l{\mapsto}\tau] \in \langle v:\tau\rangle \wedge I((\Psi[l{\mapsto}\tau]\circ\phi)[l{\mapsto}0]) * l\mapsto v \vdash I(\Psi[l{\mapsto}\tau]\circ\phi))}\ \text{Lemma A-2}}{\forall\phi,\Psi.\ (\Psi \in \langle v:\tau\rangle \wedge I(\Psi\circ\phi) * l\mapsto v \vdash I^\phi\langle l:\mathsf{ref}\ \tau\rangle)}\ (\star)}{\forall\phi.\ (I^\phi\langle v:\tau\rangle * l\mapsto v \vdash I^\phi\langle l:\mathsf{ref}\ \tau\rangle)}\ \exists\mathrm{L}}{\forall\phi.\ (I^\phi\langle v:\tau\rangle * l\mapsto v \models_1 I^\phi\langle l:\mathsf{ref}\ \tau\rangle)}\ \text{(subrelation)}}{\langle v:\tau\rangle \times l\mapsto v \models_{I*1} \langle l:\mathsf{ref}\ \tau\rangle \times emp}\ \textsc{StackComp}$$

The names of the inference rules applied above are those from the indirect triple since the corresponding ones from indirect entailment don't have a name right now. The application of StackComp is justified because $I*1 = I \succ 1$ as mentioned in Section 6.

In the step labelled $(\star)$, we exploit that the left-hand side of the entailment implies that $l \notin \mathit{supp}(\Psi\circ\phi)$. Therefore, $\Psi\circ\phi = (\Psi[l{\mapsto}\tau]\circ\phi)[l{\mapsto}0]$. Also, $\Psi \in \langle v:\tau\rangle \vdash \Psi[l{\mapsto}\tau] \in \langle v:\tau\rangle$, and we can instantiate the existential on the right of the entailment to $\Psi[l\mapsto\tau]$ since this is sure to compose with ϕ. □

A-4.2.2 Detailed proofs of Figure A-4.

Proof (of expression typings). These are simply reformulations of the rules in Figure A-3. □

Proof (of W-Weaken*).* Corollary of W-Hom-∘ (proved below). □

Proof (of W-W1 *and* W-Hom-0*).* By definition. □

Lemma A-3. $\langle v:\tau\rangle * \langle v:\tau'\rangle \vdash \tau = \tau'$.

Proof. By case analysis on v, τ, τ'. The only non-trivial case is where $v : \mathit{loc}, \tau = \mathsf{ref}\ \tau_1, \tau' = \mathsf{ref}\ \tau_1'$, where we use the fact that $[v\mapsto\tau_1]\circ[v\mapsto\tau_1']$ is only defined when $\tau_1 = \tau_1'$, and therefore $\tau = \tau'$. □

It seems somewhat coincidental that the above lemma holds in our type system, and it should not be relied on too much. It is needed to show W-Hom-∘ from right to left.

Proof (of W-Hom-∘*).* Let us first show the lemma that $[\![[x\mapsto\tau]\circ\Gamma]\!] \dashv\vdash \langle x:\tau\rangle * [\![\Gamma]\!]$. If $x \notin \mathit{supp}(\Gamma)$, then Lemma 1a applies, and the lemma follows by the trivial W-W1. If instead $x \in \mathit{supp}(\Gamma)$, let us first show the entailment

from left to right. The composition being defined implies that $\Gamma(x) = \tau$.

$$
\begin{array}{rl}
[\![[x \mapsto \tau] \circ \Gamma]\!] \dashv\vdash & \text{Lemma A-1} \\
[\![[x \mapsto \tau] \circ [x \mapsto \tau] \circ \Gamma[x \mapsto 0]]\!] \dashv\vdash & \text{(idempotence)} \\
[\![[x \mapsto \tau] \circ \Gamma[x \mapsto 0]]\!] \dashv\vdash & \text{Lemma 1a} \\
[\![[x \mapsto \tau]]\!] * [\![\Gamma[x \mapsto 0]]\!] \dashv\vdash & \text{W-w1} \\
\langle x : \tau \rangle * [\![\Gamma[x \mapsto 0]]\!] \vdash & \text{W-Dupl.} \\
\langle x : \tau \rangle * \langle x : \tau \rangle * [\![\Gamma[x \mapsto 0]]\!] \dashv\vdash & \text{Lemma 1a} \\
\langle x : \tau \rangle * [\![[x \mapsto \tau] \circ \Gamma[x \mapsto 0]]\!] \dashv\vdash & \text{Lemma A-1} \\
\langle x : \tau \rangle * [\![\Gamma]\!]. &
\end{array}
$$

In the other direction, we have to appeal to Lemma A-3.

$$
\begin{array}{rl}
\langle x : \tau \rangle * [\![\Gamma]\!] \dashv\vdash & \text{Lemma A-1} \\
\langle x : \tau \rangle * [\![[x \mapsto \Gamma(x)] \circ \Gamma[x \mapsto 0]]\!] \dashv\vdash & \text{Lemma 1a} \\
\langle x : \tau \rangle * \langle x : \Gamma(x) \rangle * [\![\Gamma[x \mapsto 0]]\!] \vdash & \text{Lemma A-3} \\
\Gamma(x) = \tau \wedge \langle x : \tau \rangle * [\![\Gamma[x \mapsto 0]]\!] \dashv\vdash & \text{Lemma 1a} \\
\Gamma(x) = \tau \wedge [\![[x \mapsto \tau] \circ \Gamma[x \mapsto 0]]\!] \dashv\vdash & \text{(idempotence)} \\
\Gamma(x) = \tau \wedge [\![[x \mapsto \tau] \circ [x \mapsto \tau] \circ \Gamma[x \mapsto 0]]\!] \vdash & \text{Lemma A-1} \\
[\![[x \mapsto \tau] \circ \Gamma]\!]. &
\end{array}
$$

With this lemma done, we can now prove W-Hom-$\circ$ by induction over $|supp(\Gamma)|$. The base case holds by Lemma 1a. In the inductive case, we get the existence of x, τ, Γ_1 such that $\Gamma = [x \mapsto \tau] \circ \Gamma_1$ with $x \notin supp(\Gamma_1)$.

$$
\begin{array}{rl}
[\![[x \mapsto \tau] \circ \Gamma_1 \circ \Gamma']\!] \dashv\vdash & \text{(above lemma)} \\
\langle x : \tau \rangle * [\![\Gamma_1 \circ \Gamma']\!] \dashv\vdash & \text{(induction hyp.)} \\
\langle x : \tau \rangle * [\![\Gamma_1]\!] * [\![\Gamma']\!] \dashv\vdash & \text{Lemma 1a} \\
[\![[x \mapsto \tau] \circ \Gamma_1]\!] * [\![\Gamma']\!]. &
\end{array}
$$

□

A-4.2.3 Detailed proofs of Figure A-5.

Proof (of W-Write'*).*

$$
\dfrac{\dfrac{\dfrac{e :_\Gamma \mathsf{ref}\ \tau \qquad e' :_\Gamma \tau}{[\![\Gamma]\!] \vdash \langle e : \mathsf{ref}\ \tau \rangle \wedge \langle e' : \tau \rangle}}{[\![\Gamma]\!] \vdash \langle e : \mathsf{ref}\ \tau \rangle * \langle e' : \tau \rangle}\,(\star) \qquad \dfrac{\dfrac{}{\vdash I.\ \{\langle e : \mathsf{ref}\ \tau \rangle * \langle e' : \tau \rangle\}\ [e] := e'\ \{\top\}}\ \text{W-Write}}{\vdash I.\ \{\langle e : \mathsf{ref}\ \tau \rangle * \langle e' : \tau \rangle * [\![\Gamma]\!]\}\ [e] := e'\ \{\top * [\![\Gamma]\!]\}}\ \textsc{Frame}}{\vdash I.\ \{[\![\Gamma]\!]\}\ [e] := e'\ \{[\![\Gamma]\!]\}}\ \text{ROC}
$$

$$
\dfrac{\vdash I.\ \{[\![\Gamma]\!]\}\ [e] := e'\ \{[\![\Gamma]\!]\}}{\vdash I * 1.\ \{[\![\Gamma]\!] \times emp\}\ [e] := e'\ \{[\![\Gamma]\!] \times emp\}}\ \textsc{ForgetL}
$$

The step labelled ($\star$) uses W-Dupl. □

Proof (of W-Read'*).*

$$\dfrac{\dfrac{e :_\Gamma \mathsf{ref}\ \tau}{[\![\Gamma]\!] \vdash \langle e : \mathsf{ref}\ \tau\rangle}\quad \dfrac{\dfrac{\dfrac{}{\vdash I.\ \{\langle e : \mathsf{ref}\ \tau\rangle\}\ x := [e]\ \{\langle x : \tau\rangle\}}\ \text{W-Read}}{\vdash I.\ \{\langle e : \mathsf{ref}\ \tau\rangle * [\![\Gamma[x \mapsto 0]]\!]\}\ x := [e]\ \{\langle x : \tau\rangle * [\![\Gamma[x \mapsto 0]]\!]\}}\ \text{Frame}}{\vdash I.\ \{\langle e : \mathsf{ref}\ \tau\rangle * [\![\Gamma]\!]\}\ x := [e]\ \{\langle x : \tau\rangle * [\![\Gamma[x \mapsto 0]]\!]\}}\ \text{W-Weaken}}{\vdash I.\ \{[\![\Gamma]\!]\}\ x := [e]\ \{[\![\Gamma[x \mapsto \tau]]\!]\}}\ \text{ROC}}{\vdash I * 1.\ \{[\![\Gamma]\!] \times emp\}\ x := [e]\ \{[\![\Gamma[x \mapsto \tau]]\!] \times emp\}}\ \text{ForgetL}$$

□

Proof (of s2w'*).*

$$\dfrac{\dfrac{e :_\Gamma \tau\quad \dfrac{\dfrac{\dfrac{\dfrac{}{\langle v : \tau\rangle \times l \mapsto v \models_{I*1} \langle l : \mathsf{ref}\ \tau\rangle \times emp}\ \text{W-s2w}}{\langle e : \tau\rangle \times x \mapsto e \models_{I*1} \langle x : \mathsf{ref}\ \tau\rangle \times emp}\ \text{(add stack)}}{[\![\Gamma[x \mapsto 0]]\!] * \langle e : \tau\rangle \times x \mapsto e \models_{I*1} [\![\Gamma[x \mapsto 0]]\!] * \langle x : \mathsf{ref}\ \tau\rangle \times emp}\ \text{Frame}}{[\![\Gamma]\!] * \langle e : \tau\rangle \times x \mapsto e \models_{I*1} [\![\Gamma[x \mapsto 0]]\!] * \langle x : \mathsf{ref}\ \tau\rangle \times emp}\ \text{W-Weaken}}{[\![\Gamma]\!] \times x \mapsto e \models_{I*1} [\![\Gamma[x \mapsto \mathsf{ref}\ \tau]]\!] \times emp}\ \text{ROC}}{\vdash I * 1.\ \{[\![\Gamma]\!] \times x \mapsto e\}\ \mathit{skip}\ \{[\![\Gamma[x \mapsto \mathsf{ref}\ \tau]]\!] \times emp\}}\ \text{(same definition)}$$

□

Proof (of W-Alloc'*).*

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash 1.\ \{emp\}\ x := \mathsf{alloc}\ 1; [x] := e\ \{x \mapsto e\}}\ \text{STD}}{\vdash I * 1.\ \{emp^{\mathsf{R}}\}\ x := \mathsf{alloc}\ 1; [x] := e\ \{(x \mapsto e)^{\mathsf{R}}\}}\ \text{ForgetR}}{\vdash I * 1.\ \{[\![\Gamma]\!] \times emp\}\ x := \mathsf{alloc}\ 1; [x] := e\ \{[\![\Gamma]\!] \times x \mapsto e\}}\ \text{Frame}\qquad e :_\Gamma \tau}{\vdash I * 1.\ \{[\![\Gamma]\!] \times emp\}\ x := \mathsf{alloc}\ 1; [x] := e\ \{[\![\Gamma[x \mapsto \mathsf{ref}\ \tau]]\!] \times emp\}}\ \text{s2w'}$$

□

Proof (of W-Assign'*).*

$$\dfrac{\dfrac{e :_\Gamma \tau\quad \dfrac{\dfrac{\dfrac{}{I.\ \{\langle e : \tau\rangle\}\ x := e\ \{\langle x : \tau\rangle\}}\ \text{Assign}}{I.\ \{\langle e : \tau\rangle * [\![\Gamma[x \mapsto 0]]\!]\}\ x := e\ \{\langle x : \tau\rangle * [\![\Gamma[x \mapsto 0]]\!]\}}\ \text{Frame}}{I.\ \{\langle e : \tau\rangle * [\![\Gamma]\!]\}\ x := e\ \{\langle x : \tau\rangle * [\![\Gamma[x \mapsto 0]]\!]\}}\ \text{W-Weaken}}{I.\ \{[\![\Gamma]\!] * [\![\Gamma]\!]\}\ x := e\ \{\langle x : \tau\rangle * [\![\Gamma[x \mapsto 0]]\!]\}}\ \text{ROC}}{\vdash I.\ \{[\![\Gamma]\!]\}\ x := e\ \{[\![\Gamma[x \mapsto \tau]]\!]\}}\ \text{W-Dupl., Lemma 1a}$$

□

Proof (of S-Write' *and* S-Free'*).*

$$\dfrac{\dfrac{\dfrac{}{1.\ \{e \mapsto \_\}\ \mathsf{free}\ e\ \{emp\}}\ \text{Free}}{I * 1.\ \{emp \times e \mapsto \_\}\ \mathsf{free}\ e\ \{emp \times emp\}}\ \text{ForgetR}}{I * 1.\ \{[\![\Gamma]\!] \times e \mapsto \_\}\ \mathsf{free}\ e\ \{[\![\Gamma]\!] \times emp\}}\ \text{Frame}$$

This proves S-Free'. The proof of S-Write' is similar. □

Proof (of S-Read', S-Alloc' *and* S-Assign'*).*

$$\dfrac{\dfrac{\dfrac{\dfrac{}{1.\ \{Q[e/x]\}\ x := e\ \{Q\}}\ \textsc{Assign}}{I * 1.\ \{emp \times Q[e/x]\}\ x := e\ \{emp \times Q\}}\ \textsc{ForgetR}}{I * 1.\ \{[\![\Gamma[x{\mapsto}0]]\!] \times Q[e/x]\}\ x := e\ \{[\![\Gamma[x{\mapsto}0]]\!] \times Q\}}\ \textsc{Frame}}{I * 1.\ \{[\![\Gamma]\!] \times Q[e/x]\}\ x := e\ \{[\![\Gamma[x{\mapsto}0]]\!] \times Q\}}\ \text{W-}\textsc{Weaken}$$

The other proofs are similar. □

Structural rules and control-flow rules follow directly from their general counterparts in fictional separation logic because of how connectives $(\wedge, \exists, *)$ distribute over the Cartesian product.

A-4.3 Fine-grained Collection

The example of a fine-grained collection is borrowed from the work on Concurrent Abstract Predicates (CAP) [DYDG$^+$10]. We reformulate it in fictional separation logic to allow comparison between this approach and the CAP approach. The two systems seem to allow very similar specifications to be exposed to clients even though the correctness proofs behind them look and feel completely different.

This example is also an opportunity to show how an existing specification can be refined in fictional separation logic. That is probably a good work flow in practice: give a correct and abstract specification of a module in standard separation logic, then show that it is refined by an indirect specification that hides sharing. These two steps can be carried out independently and concurrently as long as they agree on the intermediate specification.

In this case, we will assume a standard specification of a collection module: S_{standard} from Figure A-6. This specification is identical to the one assumed in [DYDG$^+$10], except that we have added creation and disposal functions to make the example complete and realistic.

The fine-grained specification is shown as S_{indirect} in Figure A-6. There is a lot to take in, but the idea is to have a representation predicate per (potential) member of the collection rather than one for the collection as a whole. This recognizes that if clients logically partition the set of program values *val* between them, then they can share access to the collection and ignore each others' interference since they will never query, add or remove the same values. For example, one client could promise to only access *even* numbers in a collection, while another client promised to only access *odd* numbers from the same collection. They would then access the collection with logically disjoint footprints, and the fine-grained specification would allow them to ignore each others' interference.

The intuitive reading of the four representation predicates is as follows. $In(c, v)$ means that value v is in (the collection pointed to by) c. $Out(c, v)$

$$
\begin{aligned}
S_{\text{standard}} \triangleq\ & \exists \mathit{Coll} : \mathit{loc} \times \mathcal{P}_{\mathsf{fin}}(\mathit{val}) \to \mathcal{P}(\mathit{heap}).\ \forall c, V. \\
& (\mathit{Coll}(c, \_) * \mathit{Coll}(c, \_) \vdash \bot) \wedge \\
& \{\mathit{emp}\}\ \mathsf{coll\_new}()\ \{\mathit{Coll}(\mathsf{ret}, \emptyset)\} \wedge \\
& \{\mathit{Coll}(\mathsf{c}, \_)\}\ \mathsf{coll\_free}(\mathsf{c})\ \{\mathit{emp}\} \wedge \\
& \{\mathit{Coll}(\mathsf{c}, V)\}\ \mathsf{coll\_contains}(\mathsf{c}, \mathsf{v})\ \{\mathit{Coll}(\mathsf{c}, V) \wedge \mathsf{ret} = (\mathsf{v} \in V)\} \wedge \\
& \{\mathit{Coll}(\mathsf{c}, V)\}\ \mathsf{coll\_add}(\mathsf{c}, \mathsf{v})\ \{\mathit{Coll}(\mathsf{c}, V \cup \{\mathsf{v}\})\} \wedge \\
& \{\mathit{Coll}(\mathsf{c}, V)\}\ \mathsf{coll\_remove}(\mathsf{c}, \mathsf{v})\ \{\mathit{Coll}(\mathsf{c}, V \setminus \{\mathsf{v}\})\}
\end{aligned}
$$

$$
\begin{aligned}
S_{\text{indirect}} \triangleq\ & \exists \Sigma : \mathit{sepalg}.\ \exists I : \Sigma \searrow \mathit{heap}. \\
& \exists \mathit{In} : \mathit{loc} \times \mathit{val} \to \mathcal{P}(\Sigma). \\
& \exists \mathit{Outs} : \mathit{loc} \times \mathcal{P}(\mathit{val}) \to \mathcal{P}(\Sigma). \\
& \mathsf{let}\ \mathit{Out}(c, v) := \mathit{Outs}(c, \{v\})\ \mathsf{in} \\
& \mathsf{let}\ \mathit{Own}(c, v) := \mathit{In}(c, v) \vee \mathit{Out}(c, v)\ \mathsf{in} \\
& (\forall c, v, V_1, V_2. \\
& \quad (\mathit{Own}(c, v) * \mathit{Own}(c, v) \vdash \bot) \wedge \\
& \quad (V_1 \# V_2 \Rightarrow (\mathit{Outs}(c, V_1 \cup V_2) \dashv\vdash \mathit{Outs}(c, V_1) * \mathit{Outs}(c, V_2))) \\
&) \wedge \\
& I.\ \{\mathit{emp}\}\ \mathsf{coll\_new}()\ \{\mathit{Outs}(\mathsf{ret}, \mathit{val})\} \wedge \\
& I.\ \{\exists V.\ \mathit{Outs}(\mathsf{c}, \mathit{val} \setminus V) * \forall_* v \in V.\ \mathit{In}(\mathsf{c}, v)\}\ \mathsf{coll\_free}(\mathsf{c})\ \{\mathit{emp}\} \wedge \\
& I.\ \{\mathit{In}(\mathsf{c}, \mathsf{v})\}\ \mathsf{coll\_contains}(\mathsf{c}, \mathsf{v})\ \{\mathit{In}(\mathsf{c}, \mathsf{v}) \wedge \mathsf{ret} = \mathit{true}\} \wedge \\
& I.\ \{\mathit{Out}(\mathsf{c}, \mathsf{v})\}\ \mathsf{coll\_contains}(\mathsf{c}, \mathsf{v})\ \{\mathit{Out}(\mathsf{c}, \mathsf{v}) \wedge \mathsf{ret} = \mathit{false}\} \wedge \\
& I.\ \{\mathit{Own}(\mathsf{c}, \mathsf{v})\}\ \mathsf{coll\_add}(\mathsf{c}, \mathsf{v})\ \{\mathit{In}(\mathsf{c}, \mathsf{v})\} \wedge \\
& I.\ \{\mathit{Own}(\mathsf{c}, \mathsf{v})\}\ \mathsf{coll\_remove}(\mathsf{c}, \mathsf{v})\ \{\mathit{Out}(\mathsf{c}, \mathsf{v})\}
\end{aligned}
$$

Figure A-6: Standard (coarse-grained) and indirect (fine-grained) specifications of a collection module.

means that value v is not in c. $Outs(c, V)$ generalizes this to potentially infinite sets of values. $Own(c, v)$ is the permission to access the state of v being in c without knowing whether it is currently there. In all cases, the asserter of a predicate has exclusive access to the knowledge of whether v is in c.

We can now state the correctness theorem of the fine-grained specification.

Theorem A-4. *With reference to the two specifications defined in Figure A-6, in the context of any program,* $S_{\text{standard}} \vdash S_{\text{indirect}}$.

Proof. Instantiate the existential Σ to a separation algebra where each pointer to a collection maps to a pair: the values that are surely in the collection and the values that are surely not in the collection.

$$\begin{aligned}
\Sigma &= loc \xrightarrow{\text{fin}} \Sigma_{\text{inout}} \text{ where} \\
&\quad \Sigma_{\text{inout}} \triangleq \{(V_\in, V_\notin) : \mathcal{P}_{\mathsf{fin}}(val) \times \mathcal{P}(val) \mid V_\in, V_\notin \text{ disjoint}\} \\
In(c, v) &= \{[c \mapsto (\{v\}, \emptyset)]\} \\
Outs(c, V) &= \{[c \mapsto (\emptyset, V)]\}
\end{aligned}$$

Composition in Σ_{inout} is pairwise disjoint union where possible:

$$(V_\in, V_\notin) \circ (V'_\in, V'_\notin) = \begin{cases} (V_\in \cup V'_\in, V_\notin \cup V'_\notin) & \text{if } V_\in, V_\notin, V'_\in, V'_\notin \text{ disjoint} \\ \text{undefined} & \text{otherwise} \end{cases}$$

The interpretation I is similar to previous examples: it requires all potential values to be accounted for, and then it asserts the *Coll* predicate for the values in the collection

$$I(f) = \forall_* c \in supp(f).\ (\pi_1 f(c) \cup \pi_2 f(c)) = val \wedge Coll(c, \pi_1 f(c))$$

Having chosen the existentials, we proceed to prove each conjunct of S_{indirect}.

Proof of $Own(c, v) * Own(c, v) \vdash \bot$**.** Since *Own* is defined to be a disjunction, we can consider all four possible cases and see that our composition operation is undefined in all of them.

Proof of $Outs(c, V_1 \uplus V_2) \dashv\vdash Outs(c, V_1) * Outs(c, V_2)$**.** Expanding the definition of *Outs*, this is equivalent to

$$\{[c \mapsto (\emptyset, V_1 \uplus V_2)]\} = \{[c \mapsto (\emptyset, V_1)]\} * \{[c \mapsto (\emptyset, V_2)]\}$$

Expanding the definitions of separating conjunction and of composition on finite maps, this holds if $(\emptyset, V_1 \uplus V_2) = (\emptyset, V_1) \circ (\emptyset, V_2)$, which is true by definition of composition in our SA.

Proof of coll_new**.**

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\{emp\}\ \mathsf{coll\_new}()\ \{Coll(\mathsf{ret},\emptyset)\}}\ \text{(assumption)}}{\{emp\}\ \mathsf{coll\_new}()\ \{\emptyset \cup val = val \wedge Coll(\mathsf{ret},\emptyset)\}}\ \text{(simplify)}}{\{emp\}\ \mathsf{coll\_new}()\ \{I([\mathsf{ret} \mapsto (\emptyset, val)])\}}\ \text{(definition)}}{\forall \phi.\ \{I(\phi)\}\ \mathsf{coll\_new}()\ \{I([\mathsf{ret} \mapsto (\emptyset, val)]) * I(\phi)\}}\ \textsc{Frame}}{\forall \phi.\ \{I(\phi)\}\ \mathsf{coll\_new}()\ \{I([\mathsf{ret} \mapsto (\emptyset, val)]) \circ \phi)\}}\ \text{Lemma 1b}}{I.\ \{\{0\}\}\ \mathsf{coll\_new}()\ \{\{[\mathsf{ret} \mapsto (\emptyset, val)]\}\}}\ \textsc{Enter1}}{I.\ \{emp\}\ \mathsf{coll\_new}()\ \{Outs(\mathsf{ret}, val)\}}\ \text{(definition)}$$

Proof of coll_free**.**

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\forall \phi.\ \{Coll(\mathsf{c}, V)\}\ \mathsf{coll\_free}(\mathsf{c})\ \{emp\}}\ \text{(assumption)}}{\forall \phi.\ \{V \cup (val \setminus V) = val \wedge Coll(\mathsf{c}, V)\}\ \mathsf{coll\_free}(\mathsf{c})\ \{emp\}}\ \text{(simplify)}}{\forall \phi.\ \{I([\mathsf{c} \mapsto (V, val \setminus V)])\}\ \mathsf{coll\_free}(\mathsf{c})\ \{emp\}}\ \text{(definition)}}{\forall \phi.\ \{I([\mathsf{c} \mapsto (V, val \setminus V)]) * I(\phi)\}\ \mathsf{coll\_free}(\mathsf{c})\ \{I(\phi)\}}\ \textsc{Frame}}{\forall \phi.\ \{I([\mathsf{c} \mapsto (V, val \setminus V)] \circ \phi)\}\ \mathsf{coll\_free}(\mathsf{c})\ \{I(\phi)\}}\ \text{Lemma 1a}}{I.\ \{\{[\mathsf{c} \mapsto (V, val \setminus V])\}\}\ \mathsf{coll\_free}(\mathsf{c})\ \{emp\}}\ \textsc{Enter1}}{I.\ \{\{[\mathsf{c} \mapsto (\emptyset, val \setminus V])\}\} * \{[\mathsf{c} \mapsto (V, \emptyset)]\}\}\ \mathsf{coll\_free}(\mathsf{c})\ \{emp\}}$$
$$I.\ \{Outs(\mathsf{c}, val \setminus V) * \forall_* v \in V.\ In(\mathsf{c}, v)\}\ \mathsf{coll\_free}(\mathsf{c})\ \{emp\}$$

At the application of Lemma 1a, its side condition of disjoint supports is satisfied since Σ was defined such that only the unit value composes with $(V, val \setminus V)$, so the pointer c cannot be in the support of the frame ϕ at that point.

Proof of coll_contains**, case** *In***.** In this proof, we abbreviate coll_contains as cc.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\forall V_\in.\ \{Coll(\mathsf{c}, \{\mathsf{v}\} \uplus V_\in)\}\ \mathsf{cc}(\mathsf{c}, \mathsf{v})\ \{Coll(\mathsf{c}, \{\mathsf{v}\} \uplus V_\in) \wedge \mathsf{ret} = true\}}\ \text{(assumption)}}{\begin{array}{l}\forall V_\in, V_\notin.\ \{\mathsf{v}\} \uplus V_\in \uplus V_\notin = val \Rightarrow \\ \{Coll(\mathsf{c}, \{\mathsf{v}\} \uplus V_\in)\}\ \mathsf{cc}(\mathsf{c}, \mathsf{v})\ \{Coll(\mathsf{c}, \{\mathsf{v}\} \uplus V_\in) \wedge \mathsf{ret} = true\}\end{array}}\ \text{(weaken)}}{\forall V_\in, V_\notin.\ \{\mathsf{v}\}, V_\in, V_\notin \text{ disjoint} \Rightarrow \{I([\mathsf{c} \mapsto (\{\mathsf{v}\} \uplus V_\in, V_\notin)])\}}\ \text{(simplify)}}{\begin{array}{l}\mathsf{cc}(\mathsf{c}, \mathsf{v})\ \{I([\mathsf{c} \mapsto (\{\mathsf{v}\} \uplus V_\in, V_\notin)]) \wedge \mathsf{ret} = true\}\end{array}}\ \text{Lemma 1c}}{I.\ \{[\mathsf{c} \mapsto (\{\mathsf{v}\}, \emptyset)]\}\ \mathsf{cc}(\mathsf{c}, \mathsf{v})\ \{[\mathsf{c} \mapsto (\{\mathsf{v}\}, \emptyset)] \wedge \mathsf{ret} = true\}}\ \text{(definition)}$$
$$I.\ \{In(\mathsf{c}, \mathsf{v})\}\ \mathsf{cc}(\mathsf{c}, \mathsf{v})\ \{In(\mathsf{c}, \mathsf{v}) \wedge \mathsf{ret} = true\}$$

Proof of coll_contains**, case** *Out***.** In this proof, we abbreviate coll_contains as cc.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\forall V_\in.\ \{\mathsf{v}\} \notin V_\in \Rightarrow \{Coll(\mathsf{c}, V_\in)\}\ \mathsf{cc}(\mathsf{c},\mathsf{v})\ \{Coll(\mathsf{c}, V_\in) \wedge \mathsf{ret} = false\}}\ \text{(assumption)}}{\begin{array}{c}\forall V_\in, V_\notin.\ \{\mathsf{v}\} \uplus V_\in \uplus V_\notin = val \Rightarrow \\ \{Coll(\mathsf{c}, V_\in)\}\ \mathsf{cc}(\mathsf{c},\mathsf{v})\ \{Coll(\mathsf{c}, V_\in) \wedge \mathsf{ret} = false\}\end{array}}\ \text{(weaken)}}{\begin{array}{c}\forall V_\in, V_\notin.\ \{\mathsf{v}\}, V_\in, V_\notin \text{ disjoint} \Rightarrow \{I([\mathsf{c} \mapsto (V_\in, \{\mathsf{v}\} \uplus V_\notin)])\} \\ \mathsf{cc}(\mathsf{c},\mathsf{v})\ \{I([\mathsf{c} \mapsto (V_\in, \{\mathsf{v}\} \uplus V_\notin)]) \wedge \mathsf{ret} = false\}\end{array}}\ \text{(simplify)}}{\begin{array}{c}I.\ \{[\mathsf{c} \mapsto (\emptyset, \{\mathsf{v}\})]\}\ \mathsf{cc}(\mathsf{c},\mathsf{v})\ \{[\mathsf{c} \mapsto (\emptyset, \{\mathsf{v}\})] \wedge \mathsf{ret} = false\}\end{array}}\ \begin{array}{c}\text{Lemma 1c} \\ \text{(definition)}\end{array}$$
$$I.\ \{Out(\mathsf{c},\mathsf{v})\}\ \mathsf{cc}(\mathsf{c},\mathsf{v})\ \{Out(\mathsf{c},\mathsf{v}) \wedge \mathsf{ret} = false\}$$

The add and remove functions are similar, also using Lemma 1c. □

A-4.3.1 Comparison to Concurrent Abstract Predicates.

The only essential differences from the CAP specification [DYDG$^+$10] is our *Outs* predicate, which is needed to specify creation and disposal. We have specified *Out* in terms of *Outs*. It might be tempting to do it the other way around and let *Out* be the primitive predicate and define $Outs(c, V)$ as $\forall_* v \in V.\ Out(c, v)$, but this is impossible since iterated separating conjunction is only defined on finite domains. Defining it infinitely would require an infinite composition operator on the underlying separation algebra.

A-4.4 Permission Scaling

We can easily add a convenient feature to the assertion logic of fractional permissions from Section 5.1: *permission scaling*. The idea is to have an assertion $z \cdot P$ that loosely speaking multiplies every permission in P by z. Assuming we have an operation $z \cdot h$ that multiplies every permission in the fractional heap h by z, define

$$z \cdot P \triangleq \{z \cdot h \mid h \in P\}$$

A crucial difference between this definition and the intuition mentioned is that we have

$$\frac{}{z \cdot (l_1 \overset{1}{\mapsto} v_1 * l_2 \overset{1}{\mapsto} v_2) \vdash l_1 \neq l_2}$$

for any z, while a scaling on the individual points-to predicates only admits the weaker and much stranger rule

$$\frac{z > \frac{1}{2}}{l_1 \overset{z}{\mapsto} v_1 * l_2 \overset{z}{\mapsto} v_2 \vdash l_1 \neq l_2}$$

This leads to the problem noticed by Bornat et al. [BCOP05] that a tree predicate in which every points-to assertion is multiplied by some z actually describes a DAG if $z \leq \frac{1}{2}$.

With permission scaling, one just defines an ordinary tree predicate $\mathit{tree}(t, \tau)$, and a scaled tree is then $z \cdot \mathit{tree}(t, \tau)$. The latter is a tree, not a DAG, and it is unnecessary for the tree library to export lemmas about how the predicate can be fractionally split and joined since these follow from the general theory of permission scaling. Note that one does not get the tree property from Boyland's definition of permission scaling [Boy07] because he allows fractions greater than 1 in intermediate heaps.

The following assertion logic inference rules are valid.

$$\frac{p \text{ pure}}{z{\cdot}p \dashv\vdash p} \qquad \frac{}{l \overset{z \cdot z'}{\longmapsto} v \dashv\vdash z \cdot (l \overset{z'}{\mapsto} v)} \qquad \frac{}{z \cdot (P * Q) \vdash z{\cdot}P * z{\cdot}Q}$$

$$\frac{}{(z_1 \dot{+} z_2) \cdot P \vdash z_1{\cdot}P * z_2{\cdot}P} \qquad \frac{P \text{ precise}}{z_1{\cdot}P * z_2{\cdot}P \vdash (z_1 \dot{+} z_2) \cdot P}$$

$$\frac{P \vdash Q}{z{\cdot}P \vdash z{\cdot}Q} \qquad \frac{}{z \cdot \exists x.\ P(x) \dashv\vdash \exists x.\ z \cdot P(x)}$$

The reason for introducing permission scaling here is to show that it can be added to this logic at no cost. It is reasonable to require this feature to be supported when encoding fractional permissions in other systems, such as Concurrent Abstract Predicates [DYDG$^+$10].

A-4.5 Better Monotonic Counters

The monotonic counters example in Section 3.3 was designed to address the verification challenge posed in [PP11]. With a few changes, we can make it even better. First, we can strengthen the specification of mc_read so it becomes

$$\forall i.\ I.\ \{MC(\mathsf{c}, i)\}\ \mathsf{mc\_read}(\mathsf{c})\ \{MC(\mathsf{c}, \mathsf{ret}) \wedge \mathsf{ret} \geq i\}.$$

The postcondition previously had $MC(\mathsf{c}, i)$ instead of $MC(\mathsf{c}, \mathsf{ret})$. Second, we can remove the awkward $\top$ from the weakening corollary and instead show

$$i \leq j \wedge MC(c, j) \vdash MC(c, i).$$

To make this work, we only have to change the definition of MC, leaving the other existentials as they were:

$$\begin{aligned} \Sigma &= \mathit{loc} \overset{\text{fin}}{\Rightarrow} \mathbb{Z}_\perp \text{ where composition in } \mathbb{Z} \text{ is } \mathit{max} \\ I(f) &= \forall_* c \in \mathit{supp}(f).\ \exists k \geq f(c).\ c \mapsto k \\ MC(c, i) &= \exists j \geq i.\ \{[c \mapsto j]\} \end{aligned}$$

We then have to reverify all functions with the new definition of MC. The interesting case is mc_read.

Proof of mc_read. Let $C = (\mathsf{x} := [\mathsf{c}])$, i.e., the body of mc_read.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\forall k.\ \{\mathsf{c} \mapsto k\}\ \mathsf{x} := [\mathsf{c}]\ \{\mathsf{c} \mapsto k \wedge \mathsf{x} = k\}}\ \textsc{Read}}{\forall j \geq i.\ \forall \phi.\ \forall k \geq j \circ \phi.\ \{\mathsf{c} \mapsto k\}\ C\ \{\mathsf{c} \mapsto k \wedge \mathsf{x} = k\}}\ (\star)}{\forall j \geq i.\ \forall \phi.\ \forall k \geq j \circ \phi.\ \{\mathsf{c} \mapsto k\}\ C\ \{\mathsf{x} \geq \mathsf{x} \circ \phi \wedge \mathsf{c} \mapsto \mathsf{x} \wedge \mathsf{x} \geq i\}}\ \textsc{Consequence}}{\forall j \geq i.\ \forall \phi.\ \forall k \geq j \circ \phi.\ \{\mathsf{c} \mapsto k\}\ C\ \{\exists k \geq \mathsf{x} \circ \phi.\ \mathsf{c} \mapsto k \wedge \mathsf{x} \geq i\}}\ \text{(instantiate)}}{\forall j \geq i.\ \forall \phi.\ \{\exists k \geq j \circ \phi.\ \mathsf{c} \mapsto k\}\ C\ \{\exists k \geq \mathsf{x} \circ \phi.\ \mathsf{c} \mapsto k \wedge \mathsf{x} \geq i\}}\ \textsc{Exists}}{\forall j \geq i.\ \forall \phi.\ \{I([\mathsf{c} \mapsto j \circ \phi])\}\ C\ \{I([\mathsf{c} \mapsto \mathsf{x} \circ \phi]) \wedge \mathsf{x} \geq i\}}\ \text{(definition)}}{\forall j \geq i.\ I.\ \{\{[\mathsf{c} \mapsto j]\}\}\ C\ \{\{[\mathsf{c} \mapsto \mathsf{x}]\} \wedge \mathsf{x} \geq i\}}\ \text{Lemma 1c}}{\forall j \geq i.\ I.\ \{\{[\mathsf{c} \mapsto j]\}\}\ C\ \{\exists j \geq \mathsf{x}.\ \{[\mathsf{c} \mapsto j]\} \wedge \mathsf{x} \geq i\}}\ \text{(instantiate)}}{I.\ \{\exists j \geq i.\ \{[\mathsf{c} \mapsto j]\}\}\ C\ \{\exists j \geq \mathsf{x}.\ \{[\mathsf{c} \mapsto j]\} \wedge \mathsf{x} \geq i\}}\ \textsc{Exists}}{I.\ \{MC(\mathsf{c}, i)\}\ C\ \{MC(\mathsf{c}, \mathsf{x}) \wedge \mathsf{x} \geq i\}}\ \text{(definition)}$$

The proof tree labelled $(\star)$ above should be a proof of the arithmetic fact that if $j \geq i$ and $k \geq j \circ \phi$, then $\mathsf{c} \mapsto k \wedge \mathsf{x} = k \ \vdash\ \mathsf{x} \geq \mathsf{x} \circ \phi \wedge \mathsf{c} \mapsto \mathsf{x} \wedge \mathsf{x} \geq i$.

Note that while the above i, j, k belong to $\mathbb{Z}$, the frame ϕ belongs to $\mathbb{Z}_\perp$; i.e., the integers extended with a unit element. Since we made $\mathbb{Z}$ a permission algebra with composition max, composition in $\mathbb{Z}_\perp$ behaves like max lifted to also being defined for negative infinity, i.e., the unit of $\mathbb{Z}_\perp$. Therefore, the composition $(\circ)$ above is max lifted to treat the unit as negative infinity.

Proof of mc_new. Let $C = (\mathsf{c} := \mathsf{alloc}\ 1;\ [\mathsf{c}] := 0)$, i.e., the body of mc_new.

$$\dfrac{\dfrac{\dfrac{\vdots}{I.\ \{emp\}\ C\ \{\{[\mathsf{c} \mapsto 0]\}\}}}{I.\ \{emp\}\ C\ \{\exists j \geq 0.\ \{[\mathsf{c} \mapsto j]\}\}}\ \text{(instantiate)}}{I.\ \{emp\}\ C\ \{MC(\mathsf{c}, 0)\}}\ \text{(definition)}$$

The top part of the proof works like the original proof in Section A-2.3.

Proof of mc_inc. Let $C = (\mathsf{x} := [\mathsf{c}];\ [\mathsf{c}] := \mathsf{x}{+}1)$, i.e., the body of mc_inc.

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\forall j \geq i.\ I.\ \{\{[\mathsf{c} \mapsto j]\}\}\ C\ \{\{[\mathsf{c} \mapsto j+1]\}\}}}{\forall j \geq i.\ I.\ \{\{[\mathsf{c} \mapsto j]\}\}\ C\ \{\exists j \geq i+1.\ \{[\mathsf{c} \mapsto j]\}\}}\ \text{(instantiate)}}{I.\ \{\exists j \geq i.\ \{[\mathsf{c} \mapsto j]\}\}\ C\ \{\exists j \geq i+1.\ \{[\mathsf{c} \mapsto j]\}\}}\ \textsc{Exists}}{I.\ \{MC(\mathsf{c}, i)\}\ C\ \{MC(\mathsf{c}, i+1)\}}\ \text{(definition)}$$

The top part of the proof works like the original proof in Section A-2.3.

References

[BCOP05] R. Bornat, C. Calcagno, P. W. O'Hearn, and M. J. Parkinson. Permission accounting in separation logic. In *Proceedings of POPL*, pages 259–270, 2005.

[Boy07] John Boyland. Semantics of fractional permissions with nesting. Technical Report CS-07-01, University of Wisconsin-Milwaukee, Dept. of EE & CS, December 2007.

[DYDG+10] Thomas Dinsdale-Young, Mike Dodds, Philippa Gardner, Matthew Parkinson, and Viktor Vafeiadis. Concurrent abstract predicates. In *Proceedings of ECOOP*, 2010.

[DYGW11] Thomas Dinsdale-Young, Philippa Gardner, and Mark Wheelhouse. Abstraction and refinement for local reasoning, February 2011. Journal submission.

[GBC11] Alexey Gotsman, Josh Berdine, and Byron Cook. Precision and the conjunction rule in concurrent separation logic. In *Proceedings of MFPS*, 2011.

[O'H07] Peter W. O'Hearn. Resources, concurrency and local reasoning. *Theor. Comput. Sci.*, 375(1-3):271–307, 2007.

[PP11] Alexandre Pilkiewicz and François Pottier. The essence of monotonic state. In *Proceedings of TLDI*, 2011.

[TSFC09] Gang Tan, Zhong Shao, Xinyu Feng, and Hongxu Cai. Weak updates and separation logic. In *Proceedings of APLAS*, 2009.