GSep

John Wickerson

1 Separation Logic with Modules and Abstract Predicates

Let x range over the set $\mathbb{LV}$ of logical variables, and X range over the set $\mathbb{PV}$ of program variables. Let v range over a set Val of values. Let C range over a language Com of commands.

$$C ::= C; C \mid \langle C \rangle \mid C + C \mid C^* \mid c \mid \texttt{let } (k_i = C_i)_{i \in n} \texttt{ in } C \mid \texttt{call } k$$

Let k range over a set $\mathbb{K}$ of procedure identifiers. Let K range over the set

$$\mathsf{ProcImps} \stackrel{\text{def}}{=} \mathbb{K} \to \mathsf{Com}$$

of procedure implementations that associate each procedure identifier with a closed command. We have a small-step operational semantics for commands.

$$(C, \sigma) \to_K (C', \sigma') \qquad\qquad (C, \sigma) \to_K \mathit{fault}$$

Let s range over the set $\mathsf{Store} \stackrel{\text{def}}{=} \mathbb{PV} \to \mathsf{Val}$ of mappings from program variables to values. Let h range over the set $\mathsf{Heap} \stackrel{\text{def}}{=} \mathbb{N} \rightharpoonup \mathsf{Val}$ of partial mappings from heap locations (natural numbers) to values. Let α range over a set $\mathbb{A}$ of abstract predicate identifiers. Let Π range over the set

$$\mathsf{PredEnv} \stackrel{\text{def}}{=} \mathbb{A} \to \mathcal{P}(\mathsf{Store} \times \mathsf{Heap})$$

of predicate environments that associate each abstract predicate with a set of states that it satisfies. Let P, Q range over a language Assn of assertions.

$$P ::= P * P \mid \alpha \mid \ldots$$

There exists a forcing relation $s, h \models_\Pi P$ that describes when a state (s, h) satisfies an assertion P which may refer to abstract predicates defined in Π.

$$s, h \models_\Pi P * Q \stackrel{\text{def}}{=} \exists h_0, h_1.\, h = h_0 \uplus h_1 \wedge s, h_0 \models_\Pi P \wedge s, h_1 \models_\Pi Q \qquad\qquad s, h \models_\Pi \alpha \stackrel{\text{def}}{=} (s, h) \in \Pi(\alpha)$$

Let Δ range over the set $\mathsf{PredDict} = \mathbb{A} \rightharpoonup \mathsf{Assn}$ of abstract predicate dictionaries that associate some abstract predicates with an assertion. We write $\Pi \models \Delta$ to mean

$$\forall(\alpha \mapsto P) \in \Delta.\, \Pi(\alpha) = \{(s,h) \mid s,h \models_\Pi P\}.$$

We write $\Delta \models P \Rightarrow Q$ to mean

$$\forall \Pi \models \Delta.\, \forall s,h.\, (s,h) \models_\Pi P \Rightarrow (s,h) \models_\Pi Q.$$

Let Γ range over the set

$$\mathsf{ProcSpecs} \stackrel{\text{def}}{=} \mathbb{K} \rightharpoonup (\mathsf{Assn} \times \mathsf{Assn})$$

of procedure specifications that associate some procedure identifiers with a pre- and postcondition.

Definition 1. $\mathit{safe}(Q,K,\Pi)$ is the largest set such that if $(C,s,h) \in \mathit{safe}(Q,K,\Pi)$ then:

- $\forall h_o, h_1.\, h \uplus h_o = h_1 \implies \neg \langle C, (s,h_1)\rangle \rightarrow_K \mathit{fault}$
- $C = \texttt{skip} \implies (s,h) \models_\Pi Q$
- $\forall C', s', h', h_o, h_1.\, h \uplus h_o = h_1 \wedge \langle C, (s,h_1)\rangle \rightarrow_K \langle C', (s', h_1')\rangle \implies \exists h'.\, h' \uplus h_o = h_1' \wedge (C', s', h') \in \mathit{safe}(Q,K,\Pi)$

Definition 2.

$$\Pi; K \models \{P\}\, C\, \{Q\} \stackrel{\text{def}}{=} \forall (s,h) \models_\Pi P.\, (C,s,h) \in \mathit{safe}(Q,K,\Pi)$$

Definition 3.

$$K \models_\Pi \Gamma \stackrel{\text{def}}{=} \forall \{P\}\, k\, \{Q\} \in \Gamma.\, \Pi; K \models \{P\}\, K\, k\, \{Q\}$$

Definition 4.

$$\Delta; \Gamma \models \{P\}\, C\, \{Q\} \stackrel{\text{def}}{=} \forall \Pi \models \Delta.\, \forall K \models_\Pi \Gamma.\, \Pi; K \models \{P\}\, C\, \{Q\}$$

2 GSep

2.1 Extension to semantic model

Define $X \otimes Y \stackrel{\text{def}}{=} \{(x,y) \mid x \perp y\}$. Let π range over the set

$$\mathsf{GPredEnv} \stackrel{\text{def}}{=} \mathbb{A} \rightarrow \mathcal{P}(\mathsf{Store} \times (\mathsf{Heap} \otimes \mathsf{Heap}))$$

of GSep predicate environments that associate each abstract predicate with a set of GSep states it satisfies. If $\pi \in \mathsf{GPredEnv}$ then $loc(\pi)$ and $sha(\pi)$ are elements of $\mathsf{PredEnv}$ such that

$$loc(\pi)(\alpha) = \{(s, h_L) \mid \exists h_S.\, (s, h_L, h_S) \in \pi(\alpha)\} \qquad sha(\pi)(\alpha) = \{(s, h_S) \mid (s, \emptyset, h_S) \in \pi(\alpha)\}$$

and if $\Pi \in \mathsf{PredEnv}$ then $lift(\Pi)$ is an element of $\mathsf{GPredEnv}$ such that

$$lift(\Pi)(\alpha) = \{(s, h_L, h_S) \mid h_S \perp h_L \wedge (s, h_L) \in \Pi(\alpha)\}.$$

Lemma 5. *$\pi(\alpha) \subseteq lift(loc(\pi))(\alpha)$ for all α*

Proof. Fix arbitrary α, s, h_L and h_S.

$$\begin{array}{rcl} (s, h_L, h_S) \in lift(loc(\pi))(\alpha) & = & h_S \perp h_L \wedge (s, h_L) \in loc(\pi)(\alpha) \\ & = & h_S \perp h_L \wedge (\exists h_S.\, (s, h_L, h_S) \in \pi(\alpha)) \\ & \Leftarrow & (s, h_L, h_S) \in \pi(\alpha) \end{array}$$

$\square$

Lemma 6. $loc(lift(\Pi)) = \Pi$

Proof. Fix arbitrary α, s and h_L.

$$\begin{array}{rcl} (s, h_L) \in loc(lift(\Pi))(\alpha) & = & \exists h_S.\, (s, h_L, h_S) \in lift(\Pi)(\alpha) \\ & = & \exists h_S.\, h_S \perp h_L \wedge (s, h_L) \in \Pi(\alpha) \\ & = & (s, h_L) \in \Pi(\alpha) \end{array}$$

$\square$

Let G range over the set Guar of binary relations on Heap. Let p, q range over a language GAssn of GSep assertions.

$$p ::= \boxed{P} \mid P \mid p * p \mid \underline{\alpha} \mid \ldots$$

There exists a forcing relation $s, h_L, h_S \models_\pi p$ that describes when a GSep state $(s, (h_L, h_S))$ satisfies a GSep assertion p which may refer to abstract predicates defined in π.

$$s, h_L, h_S \models_\pi P \overset{\text{def}}{=} s, h_L \models_{loc(\pi)} P \qquad s, h_L, h_S \models_\pi \boxed{P} \overset{\text{def}}{=} s, h_S \models_{sha(\pi)} P \wedge h_L = \emptyset$$

$$s, h_L, h_S \models_\pi p * q \overset{\text{def}}{=} \exists h'_L, h''_L.\, h_L = h'_L \uplus h''_L \wedge s, h'_L, h_S \models_\pi p \wedge s, h''_L, h_S \models_\pi q \qquad s, h_L, h_S \models_\pi \underline{\alpha} \overset{\text{def}}{=} (s, h_L, h_S) \in \pi(\alpha)$$

Let δ range over the set $\mathsf{GPredDict} = \mathbb{A} \rightharpoonup \mathsf{GAssn}$ of abstract predicate dictionaries that associate some abstract predicates with a GSep assertion. Note that every $\mathsf{PredDict}$ is a $\mathsf{GPredDict}$. We write $\pi \models \delta$ to mean

$$\forall(\alpha \mapsto p) \in \delta.\, \pi(\alpha) = \{(s, h_L, h_S) \mid s, h_L, h_S \models_\pi p\}.$$

Lemma 7. $\Pi \models \Delta$ *iff* $\mathit{lift}(\Pi) \models \Delta$.

Proof.

$$\begin{array}{rcl}
\mathit{lift}(\Pi) \models \Delta & = & \forall(\alpha \mapsto P) \in \Delta.\, \mathit{lift}(\Pi)(\alpha) = \{(s, h_L, h_S) \mid s, h_L, h_S \models_{\mathit{lift}(\Pi)} P\} \\
& = & \forall(\alpha \mapsto P) \in \Delta.\, \forall s, h_L, h_S.\, (h_S \perp h_L \land (s, h_L) \in \Pi(\alpha)) \Leftrightarrow s, h_L, h_S \models_{\mathit{lift}(\Pi)} P \\
& = & \forall(\alpha \mapsto P) \in \Delta.\, \forall s, h_L, h_S.\, (h_S \perp h_L \land (s, h_L) \in \Pi(\alpha)) \Leftrightarrow s, h_L \models_{\mathit{loc}(\mathit{lift}(\Pi))} P \\
& = & \forall(\alpha \mapsto P) \in \Delta.\, \forall s, h_L, h_S.\, (h_S \perp h_L \land (s, h_L) \in \Pi(\alpha)) \Leftrightarrow s, h_L \models_\Pi P \\
& = & \forall(\alpha \mapsto P) \in \Delta.\, \forall s, h_L.\, (\exists h_S.\, h_S \perp h_L \land (s, h_L) \in \Pi(\alpha)) \Leftrightarrow s, h_L \models_\Pi P \\
& = & \forall(\alpha \mapsto P) \in \Delta.\, \forall s, h_L.\, (s, h_L) \in \Pi(\alpha) \Leftrightarrow s, h_L \models_\Pi P \\
& = & \Pi \models \Delta
\end{array}$$

□

Lemma 8. $\pi \models \Delta$ *iff* $\mathit{loc}(\pi) \models \Delta$.

Proof.

$$\begin{array}{rcl}
\mathit{loc}(\pi) \models \Delta & = & \forall(\alpha \mapsto P) \in \Delta.\, \forall s, h_L.\, (s, h_L) \in \mathit{loc}(\pi)(\alpha) \Leftrightarrow s, h_L \models_{\mathit{loc}(\pi)} P \\
& = & \forall(\alpha \mapsto P) \in \Delta.\, \forall s, h_L.\, (s, h_L) \in \mathit{loc}(\pi)(\alpha) \Leftrightarrow s, h_L \models_{\mathit{loc}(\pi)} P \\
& = & \forall(\alpha \mapsto P) \in \Delta.\, \forall s, h_L.\, (\exists h_S.\, (s, h_L, h_S) \in \pi(\alpha)) \Leftrightarrow s, h_L \models_{\mathit{loc}(\pi)} P \\
& = & \forall(\alpha \mapsto P) \in \Delta.\, \forall s, h_L, h_S.\, (s, h_L, h_S) \in \pi(\alpha) \Leftrightarrow s, h_L \models_{\mathit{loc}(\pi)} P \\
& = & \forall(\alpha \mapsto P) \in \Delta.\, \pi(\alpha) = \{(s, h_L, h_S) \mid s, h_L \models_{\mathit{loc}(\pi)} P\} \\
& = & \forall(\alpha \mapsto P) \in \Delta.\, \pi(\alpha) = \{(s, h_L, h_S) \mid s, h_L, h_S \models_\pi P\}
\end{array}$$

□

We write $\delta \models_{\mathsf{G}} p \Rightarrow q$ to mean

$$\forall \pi \models \delta.\, \forall s, h_L, h_S.\, (s, h_L, h_S) \models_\pi p \Rightarrow (s, h_L, h_S) \models_\pi q.$$

Lemma 9. *In the case where p and q are local assertions and δ is a local predicate dictionary, we have:*

$$(\Delta \models_{\mathsf{G}} P \Rightarrow Q) = (\Delta \models P \Rightarrow Q)$$

Proof.

$$\begin{array}{rcl} \Delta \models_{\mathsf{G}} P \Rightarrow Q & = & \forall \pi.\, \pi \models \Delta \implies \forall s, h_L, h_S.\, (s, h_L, h_S) \models_\pi P \Rightarrow (s, h_L, h_S) \models_\pi Q \\ & = & \forall \pi.\, \mathit{loc}(\pi) \models \Delta.\, \forall s, h_L, h_S.\, (s, h_L, h_S) \models_\pi P \Rightarrow (s, h_L, h_S) \models_\pi Q \\ & = & \forall \pi.\, \mathit{loc}(\pi) \models \Delta.\, \forall s, h_L, h_S.\, (s, h_L) \models_{\mathit{loc}(\pi)} P \Rightarrow (s, h_L) \models_{\mathit{loc}(\pi)} Q \\ & = & \forall \Pi.\, \Pi \models \Delta.\, \forall s, h_L, h_S.\, (s, h_L) \models_\Pi P \Rightarrow (s, h_L) \models_\Pi Q \\ & = & \Delta \models P \Rightarrow Q \end{array}$$

$\square$

As a result of the above lemma, we can drop the G subscript in the sequel. Let γ range over the set

$$\mathsf{GProcSpecs} \stackrel{\text{def}}{=} \mathbb{K} \rightharpoonup (\mathsf{GAssn} \times \mathsf{GAssn} \times \mathsf{Guar})$$

of procedure specifications that associate some procedure identifiers with a pre- and postcondition in GAssn plus a guarantee. The following function converts an Assn into an GAssn by underlining all of the abstract predicates (thus allowing their values to range over the shared state not just the local state).

$$\underline{P * Q} \stackrel{\text{def}}{=} \underline{P} * \underline{Q} \qquad\qquad \underline{\alpha} \stackrel{\text{def}}{=} \underline{\alpha} \qquad\qquad \ldots$$

We can convert elements of ProcSpecs to elements of GProcSpecs like so:

$$\Gamma[G] = \{\{P\}k\{Q\}[G] \mid \{P\}k\{Q\} \in \Gamma\}$$

Definition 10. $\mathit{safe}(q, K, \pi, G)$ is the largest set such that if $(C, s, h_L, h_S) \in \mathit{safe}(q, K, \pi, G)$ then:

- $\forall h_o, h.\, h_L \uplus h_S \uplus h_o = h \implies \neg \langle C, (s, h) \rangle \rightarrow_K \mathit{fault}$
- $C = \texttt{skip} \implies (s, h_L, h_S) \models_\pi q$
- $\forall C', s', h'_S, h'_L, h_o, h.\, h_L \uplus h_S \uplus h_o = h \wedge \langle C, (s, h) \rangle \rightarrow_K \langle C', (s', h') \rangle \implies \exists h'_L, h'_S.\, h'_L \uplus h'_S \uplus h_o = h' \wedge (h_S, h'_S) \in G \wedge (C', s', h'_L, h'_S) \in \mathit{safe}(q, K, \pi, G)$

Definition 11.
$$\pi; K \models \{p\}\,C\,\{q\}[G] \stackrel{\text{def}}{=} \forall (s, h_L, h_S) \models_\pi p.\,(C, s, h_L, h_S) \in \mathit{safe}(q, K, \pi, G)$$

Definition 12.
$$K \models_\pi \gamma \stackrel{\text{def}}{=} \forall \{p\}\,k\,\{q\}[G] \in \gamma.\,\pi; K \models \{p\}\,K\,k\,\{q\}[G]$$

Definition 13.
$$\delta; \gamma \models \{p\}\,C\,\{q\}[G] \stackrel{\text{def}}{=} \forall \pi \models \delta.\,\forall K \models_\pi \gamma.\,\pi; K \models \{p\}\,C\,\{q\}[G]$$

3 Proof rules

$$\textsc{Underline}\quad \frac{}{\Delta \models \underline{P} \Rightarrow P}$$

$$\textsc{G-Erase}\quad \frac{\Delta; \Gamma[G] \vdash \{P * \boxed{P'}\}\,C\,\{Q * \boxed{Q'}\}[G]}{\Delta; \Gamma \vdash \{P * P'\}\,C\,\{Q * Q'\}}$$

$$\textsc{G-Basic}\quad \frac{\Delta; \Gamma \vdash \{P\}\,C\,\{Q\}}{\Delta; \Gamma[G] \vdash \{P\}\,C\,\{Q\}[G]}$$

$$\textsc{Proc-Conseq}\quad \frac{\begin{array}{c}\delta \models p' \Rightarrow p \qquad \delta \models q \Rightarrow q' \qquad G \subseteq G' \\ \delta; \gamma \uplus \{p'\}k\{q'\}[G'] \vdash \{p_0\}\,C\,\{q_0\}[G_0]\end{array}}{\delta; \gamma \uplus \{p\}k\{q\}[G] \vdash \{p_0\}\,C\,\{q_0\}[G_0]}$$

$$\textsc{Conseq}\quad \frac{\begin{array}{c}\delta \models p \Rightarrow p' \qquad \delta \models q' \Rightarrow q \qquad G' \subseteq G \\ \delta; \gamma \vdash \{p'\}\,C\,\{q'\}[G']\end{array}}{\delta; \gamma \vdash \{p\}\,C\,\{q\}[G]}$$

$$\textsc{G-PredE}\quad \frac{\begin{array}{c}\delta \uplus \delta'; \Gamma \vdash \{p\}\,C\,\{q\}[G] \\ \forall \alpha \in \mathrm{dom}(\delta').\,\alpha \notin \delta, \Gamma, p, q\end{array}}{\delta; \Gamma \vdash \{p\}\,C\,\{q\}[G]}$$

$$\textsc{G-PredI}\quad \frac{\delta; \Gamma \vdash \{p\}\,C\,\{q\}[G]}{\delta \uplus \delta'; \Gamma \vdash \{p\}\,C\,\{q\}[G]}$$

$$\textsc{G-HypFrame}\quad \frac{\delta; \Gamma \uplus (\{p_i\}k_i\{q_i\}[G_i])_{i \in n} \vdash \{p\}\,C\,\{q\}[G]}{\delta; \Gamma \uplus (\{p_i * r\}k_i\{q_i * r\}[G_i])_{i \in n} \vdash \{p * r\}\,C\,\{q * r\}[G]}$$

$$\textsc{G-Let}\quad \frac{\begin{array}{c}(\delta; \gamma \vdash \{p_i\}\,C_i\,\{q_i\}[G])_{i \in n} \\ \delta; \gamma \uplus (\{p_i\}k_i\{q_i\}[G])_{i \in n} \vdash \{p\}\,C\,\{q\}[G]\end{array}}{\delta; \gamma \vdash \{p\}\,\texttt{let}\ (f_i = C_i)_{i \in n}\ \texttt{in}\ C\,\{q\}[G]}$$

The following rule....

$$\textsc{G-DerivModule}\quad \frac{\begin{array}{c}(\Delta \uplus \delta; \Gamma \vdash \{\underline{P_i} * \boxed{R}\}\,C_i\,\{\underline{Q_i} * \boxed{R}\}[G])_{i \in n} \\ \Delta; \Gamma \uplus (\{P_i\}k_i\{Q_i\})_{i \in n} \vdash \{P\}\,C\,\{Q\}\end{array}}{\Delta; \Gamma \vdash \{P * R\}\,\texttt{let}\ (f_i = C_i)_{i \in n}\ \texttt{in}\ C\,\{Q * R\}}$$

... can be derived like so:

$$
\dfrac{
(\Delta \uplus \delta; \Gamma \vdash \{\underline{P_i} * \boxed{R}\}\, C_i \,\{\underline{Q_i} * \boxed{R}\}[G])_{i\in n}
\qquad
\dfrac{
\dfrac{
\mathrm{dom}(\delta) = \bigcup_{i\in n} ap(P_i \vee Q_i)
\qquad
\dfrac{
\dfrac{
\Delta; \Gamma \uplus (\{P_i\}k_i\{Q_i\})_{i\in n} \vdash \{P\}\, C \,\{Q\}
}{
\Delta; \Gamma[G] \uplus (\{P_i\}k_i\{Q_i\}[G])_{i\in n} \vdash \{P\}\, C \,\{Q\}[G]
}\;\textsc{G-Basic}
}{
\Delta \uplus \delta; \Gamma[G] \uplus (\{P_i\}k_i\{Q_i\}[G])_{i\in n} \vdash \{P\}\, C \,\{Q\}[G]
}\;\textsc{G-PredI}
}{
\Delta \uplus \delta; \Gamma[G] \uplus (\{\underline{P_i}\}k_i\{\underline{Q_i}\}[G])_{i\in n} \vdash \{P\}\, C \,\{Q\}[G]
}\;\textsc{Proc-Conseq}
}{
\Delta \uplus \delta; \Gamma[G] \uplus (\{\underline{P_i} * \boxed{R}\}k_i\{\underline{Q_i} * \boxed{R}\}[G])_{i\in n} \vdash \{P * \boxed{R}\}\, C \,\{Q * \boxed{R}\}[G]
}\;\textsc{HypFrame}
}{
\dfrac{
\dfrac{
\Delta \uplus \delta; \Gamma[G] \vdash \{P * \boxed{R}\}\, \texttt{let}\ (f_i = C_i)_{i\in n}\ \texttt{in}\ C \,\{Q * \boxed{R}\}[G]
}{
\Delta; \Gamma[G] \vdash \{P * \boxed{R}\}\, \texttt{let}\ (f_i = C_i)_{i\in n}\ \texttt{in}\ C \,\{Q * \boxed{R}\}[G]
}\;\textsc{G-PredE}
}{
\Delta; \Gamma \vdash \{P * R\}\, \texttt{let}\ (f_i = C_i)_{i\in n}\ \texttt{in}\ C \,\{Q * R\}
}\;\textsc{G-Erase}
}\;\textsc{G-Let}
$$

4 Soundness?

4.1 Underline

Fix arbitrary $\pi \models \Delta$. Show $\forall P.\, \forall s, h_L, h_S.\, (s, h_L, h_S \models_\pi P) \Leftarrow (s, h_L, h_S \models_\pi \underline{P})$ by induction on P.

Case $P = \alpha$. Suppose $\alpha \in \mathrm{dom}(\Delta)$. Hence $(s, h_L, h_S) \in \pi(\alpha) \Leftrightarrow s, h_L, h_S \models_\pi \Delta(\alpha)$.

$$
\begin{array}{rcl}
s, h_L, h_S \models_\pi \alpha & = & h_S \perp h_L \wedge s, h_L \models_{loc(\pi)} \alpha \\
& = & h_S \perp h_L \wedge (s, h_L) \in loc(\pi)(\alpha) \\
& = & h_S \perp h_L \wedge (\exists h_S.\, (s, h_L, h_S) \in \pi(\alpha)) \\
& = & h_S \perp h_L \wedge (\exists h_S.\, s, h_L, h_S \models_\pi \Delta(\alpha)) \\
& = & h_S \perp h_L \wedge (\exists h_S.\, s, h_L \models_{loc(\pi)} \Delta(\alpha)) \\
& = & h_S \perp h_L \wedge s, h_L \models_{loc(\pi)} \Delta(\alpha) \\
& = & s, h_L, h_S \models_\pi \Delta(\alpha) \\
& = & (s, h_L, h_S) \in \pi(\alpha) \\
& = & s, h_L, h_S \models_\pi \underline{\alpha}
\end{array}
$$

Case $P = P' * P''$.

$$\begin{array}{rcl} s, h_L, h_S \models_\pi P' * P'' & = & \exists h'_L, h''_L.\, h_L = h'_L \uplus h''_L \wedge s, h'_L, h_S \models_\pi P' \wedge s, h''_L, h_S \models_\pi P'' \\ & \Leftarrow & \exists h'_L, h''_L.\, h_L = h'_L \uplus h''_L \wedge s, h'_L, h_S \models_\pi \underline{P'} \wedge s, h''_L, h_S \models_\pi \underline{P''} \\ & = & s, h_L, h_S \models_\pi \underline{P'} * \underline{P''} \\ & = & s, h_L, h_S \models_\pi \underline{P' * P''} \end{array}$$

Lemma 14 (G-Erase 1)**.** *If* $s, h_L, h_S \models_\Pi \underline{P}$ *then* $s, h_L \models_\Pi P$.

Lemma 15 (G-Erase 2)**.** *If* $(C, s, h_L) \in \mathit{safe}(Q, K, \Pi)$ *then* $(C, s, h_L, h_S) \in \mathit{safe}(\underline{Q}, K, \Pi, G)$.

4.2 G-Erase

Suppose $\Delta \models \{P\}C\{Q\}$. Show $G; \Delta \models \{\underline{P}\}C\{Q\}$. Fix arbitrary K. No procedure environment, so K not used. Fix arbitrary π. Assume $\pi \models \Delta$. Then $\pi = \Pi$ for some Π. Assume $s, h_L, h_S \models_\Pi \underline{P}$. Then $s, h_L \models_\Pi P$ by lemma. So $(C, s, h_L) \in \mathit{safe}(Q, K, \Pi)$ by assumption. Then we need $(C, s, h_L, h_S) \in \mathit{safe}(\underline{Q}, K, \Pi, G)$ too.