1 Syntax

$program$	$::= \overline{cls}\ \overline{s}$
cls	$::= \texttt{class}\ C\ \{\overline{field}\ \overline{method}\}$
$field$	$::= T\ f;$
$method$	$::= T\ m(T\ x)\ contract\ \{\overline{s}\}$
$contract$	$::= \texttt{requires}\ \phi;\ \texttt{ensures}\ \phi;$
T	$::= \texttt{int} \mid C$
s	$::= x.f := y; \mid x := e; \mid x := \texttt{new}\ C; \mid x := y.m(z);$
	$\mid \texttt{return}\ x; \mid \texttt{assert}\ \phi; \mid \texttt{release}\ \phi; \mid T\ x;$
ϕ	$::= \texttt{true} \mid e = e \mid e \neq e \mid \texttt{acc}(e.f) \mid x : T \mid \phi * \phi$
e	$::= v \mid x \mid e.f$
x	$::= \texttt{this} \mid \texttt{result} \mid \langle other \rangle$
v	$::= o \mid n \mid \texttt{null}$
n	$\in\ \mathbb{Z}$
H	$\in\ (o \rightharpoonup (C, \overline{(f \rightharpoonup v)}))$
ρ	$\in\ (x \rightharpoonup v)$
A_s	$::= \overline{(e, f)}$
A_d	$::= \overline{(o, f)}$
S	$::= (\rho, A_d, \overline{s}) \cdot S \mid nil$

2 Assumptions

All the rules in the following sections are implicitly parameterized over a *programp* that is well-formed.

2.0.1 Well-formed program ($program$ $\texttt{OK}$)

$$\frac{\overline{cls_i\ \texttt{OK}}}{(\overline{cls_i}\ \overline{s})\ \texttt{OK}}\ \textsc{OKProgram}$$

2.0.2 Well-formed class (cls $\texttt{OK}$)

$$\frac{\text{unique } field\text{-names} \qquad \text{unique } method\text{-names} \qquad \overline{method_i\ \texttt{OK in}\ C}}{(\texttt{class}\ C\ \{\overline{field_i}\ \overline{method_i}\})\ \texttt{OK}}\ \textsc{OKClass}$$

2.0.3 Well-formed method ($method$ $\texttt{OK in}$ C)

$$\frac{\begin{array}{c} FV(\phi_1) \subseteq \{x, \texttt{this}\} \qquad FV(\phi_2) \subseteq \{x, \texttt{this}, \texttt{result}\} \\ \vdash \{x : T_x * \texttt{this} : C * \phi_1\}\overline{s}\{x : T_x * \texttt{this} : C * \texttt{result} : T_m * \phi_2\} \\ \emptyset \vdash_{\texttt{sfrm}} \phi_1 \qquad \emptyset \vdash_{\texttt{sfrm}} \phi_2 \qquad \overline{\neg writesTo(s_i, x)} \end{array}}{(T_m\ m(T_x\ x)\ \texttt{requires}\ \phi_1;\ \texttt{ensures}\ \phi_2;\ \{\overline{s}\})\ \texttt{OK in}\ C}\ \textsc{OKMethod}$$

3 Static semantics

3.1 Expressions ($A_s \vdash_{\texttt{sfrm}} e$)

$$\frac{}{A \vdash_{\texttt{sfrm}} x}\ \textsc{WFVar}$$

$$\frac{}{A \vdash_{\texttt{sfrm}} v}\ \textsc{WFValue}$$

$$\frac{(e, f) \in A \qquad A \vdash_{\texttt{sfrm}} e}{A \vdash_{\texttt{sfrm}} e.f}\ \textsc{WFField}$$

3.2 Formulas ($A_s \vdash_{\texttt{sfrm}} \phi$)

$$\frac{}{A \vdash_{\texttt{sfrm}} \texttt{true}}\ \textsc{WFTrue}$$

$$\frac{A \vdash_{\texttt{sfrm}} e_1 \qquad A \vdash_{\texttt{sfrm}} e_2}{A \vdash_{\texttt{sfrm}} (e_1 = e_2)}\ \textsc{WFEqual}$$

$$\frac{A \vdash_{\texttt{sfrm}} e_1 \qquad A \vdash_{\texttt{sfrm}} e_2}{A \vdash_{\texttt{sfrm}} (e_1 \neq e_2)}\ \textsc{WFNEqual}$$

$$\frac{A \vdash_{\texttt{sfrm}} e}{A \vdash_{\texttt{sfrm}} \texttt{acc}(e.f)}\ \textsc{WFAcc}$$

$$\frac{}{A \vdash_{\texttt{sfrm}} x : T}\ \textsc{WFType}$$

$$\frac{A_s \vdash_{\texttt{sfrm}} \phi_1 \qquad A_s \cup \lfloor \phi_1 \rfloor \vdash_{\texttt{sfrm}} \phi_2}{A_s \vdash_{\texttt{sfrm}} \phi_1 * \phi_2}\ \textsc{WFSepOp}$$

3.2.1 Implication ($\phi_1 \dot{\implies} \phi_2$)

Conservative approx. of $\phi_1 \implies \phi_2$.

3.3 Footprint ($\lfloor \phi \rfloor = A_s$)

$$
\begin{aligned}
&\lfloor \texttt{true} \rfloor && = \emptyset \\
&\lfloor e_1 = e_2 \rfloor && = \emptyset \\
&\lfloor e_1 \neq e_2 \rfloor && = \emptyset \\
&\lfloor \texttt{acc}(e.f) \rfloor && = \{(e, f)\} \\
&\lfloor \phi_1 * \phi_2 \rfloor && = \lfloor \phi_1 \rfloor \cup \lfloor \phi_2 \rfloor
\end{aligned}
$$

3.4 Type ($\phi \vdash e : T$)

$$\frac{}{\phi \vdash n : \texttt{int}}\ \textsc{STValNum}$$

$$\frac{}{\phi \vdash \texttt{null} : T}\ \textsc{STValNull}$$

$$\frac{\phi \implies (x : T)}{\phi \vdash x : T}\ \textsc{STVar}$$

$$\frac{\phi \vdash e : C \qquad \vdash C.f : T}{\phi \vdash e.f : T}\ \textsc{STField}$$

3.5 Hoare ($\vdash \{\phi\}\overline{s}\{\phi\}$)

$$\frac{\vdash \{\phi_p\}s_1\{\phi_q\} \qquad \vdash \{\phi_q\}s_2\{\phi_r\}}{\vdash \{\phi_p\}s_1; s_2\{\phi_r\}}\ \textsc{HSec}$$

$$\frac{\phi \implies \phi' \qquad \emptyset \vdash_{\texttt{sfrm}} \phi' \qquad x \notin FV(\phi') \qquad \phi \vdash x : C \qquad \texttt{fields}(C) = \overline{f}}{\vdash \{\phi\}x := \texttt{new}\ C\{\overline{\texttt{acc}(x, f_i)*}x : C * (x \neq \texttt{null}) * \phi'\}}\ \textsc{HNewObj}$$

$$\frac{\begin{array}{c}\phi \implies \texttt{acc}(x.f) * (x \neq \texttt{null}) * \phi' \\ \emptyset \vdash_{\texttt{sfrm}} \phi' \qquad \phi \vdash x : C \qquad \phi \vdash y : T \qquad \vdash C.f : T\end{array}}{\vdash \{\phi\}x.f := y\{x : C * \texttt{acc}(x.f) * (x \neq \texttt{null}) * (x.f = y) * \phi'\}}\ \textsc{HFieldAssign}$$

$$\frac{\begin{array}{c}\phi \implies \phi' \\ \emptyset \vdash_{\texttt{sfrm}} \phi' \qquad x \notin FV(\phi') \qquad x \notin FVe(e) \qquad \phi \vdash x : T \qquad \phi \vdash e : T \qquad \lfloor \phi' \rfloor \vdash_{\texttt{sfrm}} e\end{array}}{\vdash \{\phi\}x := e\{\phi' * (x = e)\}}\ \textsc{HVarAssign}$$

$$\frac{\phi \implies \phi' \qquad \emptyset \vdash_{\texttt{sfrm}} \phi' \qquad \texttt{result} \notin FV(\phi') \qquad \phi \vdash x : T \qquad \phi \vdash \texttt{result} : T}{\vdash \{\phi\}\texttt{return}\ x\{\texttt{result} : T * (\texttt{result} = x) * \phi'\}}\ \textsc{HReturn}$$

$$\frac{\begin{array}{c}\phi \vdash y : C \qquad \texttt{mmethod}(C, m) = T_r\ m(T_p\ z)\ \texttt{requires}\ \phi_{pre};\ \texttt{ensures}\ \phi_{post};\ \{\_\} \\ \phi \vdash x : T_r \qquad \phi \vdash z' : T_p \qquad \phi \implies (y \neq \texttt{null}) * \phi_p * \phi_r \\ \emptyset \vdash_{\texttt{sfrm}} \phi_r \qquad x \notin FV(\phi_r) \qquad @listDistinct(x, x \cdot y \cdot z' \cdot \emptyset) \\ \phi_p = \phi_{pre}[y, z'/\texttt{this}, z] \qquad \phi_q = \phi_{post}[y, z', x/\texttt{this}, z, \texttt{result}]\end{array}}{\vdash \{\phi\} x := y.m(z')\{\phi_q * \phi_r\}} \textsc{HApp}$$

$$\frac{\phi_1 \implies \phi_2}{\vdash \{\phi_1\}\texttt{assert}\ \phi_2\{\phi_1\}} \textsc{HAssert}$$

$$\frac{\phi_1 \implies \phi_2 * \phi_r \qquad \emptyset \vdash_{\texttt{sfrm}} \phi_r}{\vdash \{\phi_1\}\texttt{release}\ \phi_2\{\phi_r\}} \textsc{HRelease}$$

$$\frac{\phi \implies \phi' \qquad \emptyset \vdash_{\texttt{sfrm}} \phi' \qquad x \notin FV(\phi')}{\vdash \{\phi\}T\ x\{x : T * (x = \texttt{defaultValue}(T)) * \phi'\}} \textsc{HDeclare}$$

Note: issue with HApp and z' in the post-condition: the substitution reflects **any changes** made to z onto z' which is wrong in general (except we make z' a by-ref parameter in the small-step semantics)

4 Dynamic semantics

4.1 Expressions ($H, \rho \vdash e \Downarrow v$)

$$\frac{}{H, \rho \vdash x \Downarrow \rho(x)} \textsc{EEVar}$$

$$\frac{}{H, \rho \vdash v \Downarrow v} \textsc{EEValue}$$

$$\frac{H, \rho \vdash e \Downarrow o}{H, \rho \vdash e.f \Downarrow H(o)(f)} \textsc{EEAcc}$$

4.2 Formulas ($H, \rho, A \vDash \phi$)

$$\frac{}{H, \rho, A \vDash \texttt{true}} \textsc{EATrue}$$

$$\frac{H, \rho \vdash e_1 \Downarrow v_1 \qquad H, \rho \vdash e_2 \Downarrow v_2 \qquad v_1 = v_2}{H, \rho, A \vDash (e_1 = e_2)} \textsc{EAEqual}$$

$$\frac{H, \rho \vdash e_1 \Downarrow v_1 \qquad H, \rho \vdash e_2 \Downarrow v_2 \qquad v_1 \neq v_2}{H, \rho, A \vDash (e_1 \neq e_2)}\ \textsc{EANEqual}$$

$$\frac{H, \rho \vdash e \Downarrow o \qquad (o, f) \in A}{H, \rho, A \vDash \texttt{acc}(e.f)}\ \textsc{EAAcc}$$

$$\frac{\rho(x) = v \qquad H \vdash v : T}{H, \rho, A \vDash x : T}\ \textsc{EAType}$$

$$\frac{A_1 = A \backslash A_2 \qquad H, \rho, A_1 \vDash \phi_1 \qquad H, \rho, A_2 \vDash \phi_2}{H, \rho, A \vDash \phi_1 * \phi_2}\ \textsc{EASepOp}$$

We give a denotational semantics of formulas as $[\![\phi]\!] = \{\ (H, \rho, A) \mid H, \rho, A \vDash \phi\ \}$
Note: ϕ satisfiable $\iff [\![\phi]\!] \neq \emptyset$

4.2.1 Implication ($\phi_1 \implies \phi_2$)

$$\phi_1 \implies \phi_2 \qquad \iff \qquad [\![\phi_1]\!] \subseteq [\![\phi_2]\!]$$

4.3 Footprint ($\lfloor \phi \rfloor_{H,\rho} = A_d$)

$$\begin{aligned}
\lfloor \texttt{true} \rfloor_{H,\rho} &= \emptyset \\
\lfloor e_1 = e_2 \rfloor_{H,\rho} &= \emptyset \\
\lfloor e_1 \neq e_2 \rfloor_{H,\rho} &= \emptyset \\
\lfloor \texttt{acc}(x.f) \rfloor_{H,\rho} &= \{(o, f)\} \text{ where } H, \rho \vdash x \Downarrow o \\
\lfloor \phi_1 * \phi_2 \rfloor_{H,\rho} &= \lfloor \phi_1 \rfloor_{H,\rho} \cup \lfloor \phi_2 \rfloor_{H,\rho}
\end{aligned}$$

4.4 Small-step ($(H, S) \to (H, S)$)

$$\frac{H, \rho \vdash x \Downarrow o \qquad H, \rho \vdash y \Downarrow v_y \qquad (o, f) \in A \qquad H' = H[o \mapsto [f \mapsto v_y]]}{(H, (\rho, A, x.f := y; \overline{s}) \cdot S) \to (H', (\rho, A, \overline{s}) \cdot S)}\ \textsc{ESFieldAssign}$$

$$\frac{H, \rho \vdash e \Downarrow v \qquad \rho' = \rho[x \mapsto v]}{(H, (\rho, A, x := e; \overline{s}) \cdot S) \to (H, (\rho', A, \overline{s}) \cdot S)}\ \textsc{ESVarAssign}$$

$$\frac{\begin{array}{c} o \notin \texttt{dom}(H) \qquad \texttt{fields}(C) = \overline{T}\ \overline{f} \\ \rho' = \rho[x \mapsto o] \qquad A' = A * \overline{(o, f_i)} \qquad H' = H[o \mapsto \overline{[f \mapsto \texttt{defaultValue}(T)]}] \end{array}}{(H, (\rho, A, x := \texttt{ new } C; \overline{s}) \cdot S) \to (H', (\rho', A', \overline{s}) \cdot S)}\ \textsc{ESNewObj}$$

$$\frac{H, \rho \vdash x \Downarrow v_x \qquad \rho' = \rho[\texttt{result} \mapsto v_x]}{(H, (\rho, A, \texttt{return}\ x; \overline{s}) \cdot S) \rightarrow (H, (\rho', A, \overline{s}) \cdot S)}\ \textsc{ESReturn}$$

$$\frac{\begin{array}{c} H, \rho \vdash y \Downarrow o \qquad H, \rho \vdash z \Downarrow v \\ H(o) = (C, \_) \qquad \texttt{mmethod}(C, m) = T_r\ m(T\ w)\ \texttt{requires}\ \phi;\ \texttt{ensures}\ \_;\ \{\overline{r}\} \\ \rho' = [\texttt{result} \mapsto \texttt{defaultValue}(T_r), \texttt{this} \mapsto o, w \mapsto v] \qquad H, \rho', A \vDash \phi \qquad A' = \lfloor \phi \rfloor_{H,\rho'} \end{array}}{(H, (\rho, A, x := y.m(z); \overline{s}) \cdot S) \rightarrow (H, (\rho', A', \overline{r}) \cdot (\rho, A \setminus A', x := y.m(z); \overline{s}) \cdot S)}\ \textsc{ESApp}$$

$$\frac{\begin{array}{c} H, \rho \vdash y \Downarrow o \qquad H(o) = (C, \_) \\ \texttt{mpost}(C, m) = \phi \qquad H, \rho', A' \vDash \phi \qquad A'' = \lfloor \phi \rfloor_{H,\rho'} \qquad H, \rho' \vdash \texttt{result} \Downarrow v_r \end{array}}{(H, (\rho', A', \emptyset) \cdot (\rho, A, x := y.m(z); \overline{s}) \cdot S) \rightarrow (H, (\rho[x \mapsto v_r], A * A'', \overline{s}) \cdot S)}\ \textsc{ESAppFinish}$$

$$\frac{H, \rho, A \vDash \phi}{(H, (\rho, A, \texttt{assert}\ \phi; \overline{s}) \cdot S) \rightarrow (H, (\rho, A, \overline{s}) \cdot S)}\ \textsc{ESAssert}$$

$$\frac{H, \rho, A \vDash \phi \qquad A' = A \setminus \lfloor \phi \rfloor_{H,\rho}}{(H, (\rho, A, \texttt{release}\ \phi; \overline{s}) \cdot S) \rightarrow (H, (\rho, A', \overline{s}) \cdot S)}\ \textsc{ESRelease}$$

$$\frac{\rho' = \rho[x \mapsto \texttt{defaultValue}(T)]}{(H, (\rho, A, T\ x; \overline{s}) \cdot S) \rightarrow (H, (\rho', A, \overline{s}) \cdot S)}\ \textsc{ESDeclare}$$

5 Gradualization

5.1 Syntax

$$\widetilde{\phi} \quad ::= \quad \phi \mid ?\ *\ \phi$$

Note: consider ? in other positions as "self-framing delimiter", but with semantically identical meaning.

5.2 Concretization

Syntax $\hat{\phi}$:= self-framed and satisfiable ϕ

$$\begin{array}{ll} \gamma(\hat{\phi}) & = \{\ \hat{\phi}\ \} \\ \gamma(?\ *\ \phi') & = \{\ \hat{\phi} \mid \hat{\phi} \implies \phi'\ \}\ \text{ if } \phi' \text{ satisfiable} \\ \gamma(\phi) \text{ undefined otherwise} & \end{array}$$

5.3 Abstraction

$$\begin{array}{ll}
\alpha(\emptyset) \text{ undefined} & \\
\alpha(\{\ \phi\ \}) & = \phi \\
\alpha(\overline{\phi} \text{ with maximum element } \phi) & =\ ?\ *\ \phi \\
\alpha(\overline{\phi}) & =\ ? \quad \text{otherwise}
\end{array}$$

5.4 Gradual Lifting

5.4.1 Self framing

$$\frac{A \vdash_{\texttt{sfrm}} \phi}{A \mathrel{\widetilde{\vdash_{\texttt{sfrm}}}} \phi}\ \textsc{GSfrmNonGrad}$$

$$\frac{}{A \mathrel{\widetilde{\vdash_{\texttt{sfrm}}}}\ ?\ *\ \phi}\ \textsc{GSfrmGrad}$$

5.4.2 Implication

$$\frac{\phi_1 \implies \phi_2}{\phi_1 \mathrel{\widetilde{\implies}} \widetilde{\phi_2}}\ \textsc{GImplNonGrad}$$

$$\frac{\hat{\phi_m} \implies \phi_2 \qquad \hat{\phi_m} \implies \phi_1}{?\ *\ \phi_1 \mathrel{\widetilde{\implies}} \widetilde{\phi_2}}\ \textsc{GImplGrad}$$

$\hat{\phi_m}$ is evidence!

Consistent transitivity
While $\implies$ is transitive, $\widetilde{\implies}$ is generally not.
But maybe not even necessary with smarter hoare rules?

5.4.3 Hoare and evidence

Discussion/Considerations:

- The post-condition-ϕ seems to inherit its gradual-ness from implication, which itself does not care about whether its second argument is gradual or not...
- Gradual

Example:

$$\frac{x \notin FV(\widetilde{\phi}') \qquad x \notin FV(e) \qquad \dfrac{\epsilon \vdash \widetilde{\phi} \mathrel{\widetilde{\implies}} \widetilde{\phi}' \qquad \emptyset \mathrel{\widetilde{\vdash_{\texttt{sfrm}}}} \widetilde{\phi}'}{\epsilon \vdash \widetilde{\phi} \vdash x : T \qquad \epsilon \vdash \widetilde{\phi} \vdash e : T} \qquad \epsilon \vdash \lfloor \widetilde{\phi}' \rfloor \vdash_{\texttt{sfrm}} e}{\vdash \{\widetilde{\phi}\} x := e \{\widetilde{\phi}' * (x = e)\}}\ \textsc{GHVarAssign}$$

Collapsing (hidden) gradual implications into a single one:

$$\dfrac{\begin{array}{c}\epsilon \vdash \widetilde{\phi} \mathrel{\widetilde{\Longrightarrow}} (x : T) * [\![e]\!]_{acc} * [e : T] * \widetilde{\phi}' \\ \emptyset \vdash_{\texttt{sfrm}} \widetilde{\phi}' \qquad x \notin FV(\widetilde{\phi}') \qquad x \notin FV(e) \qquad [e : T]\end{array}}{\vdash \{\widetilde{\phi}\} x := e \{[\![e]\!]_{acc} * \widetilde{\phi}' * (x = e)\}} \textsc{GHVarAssign}$$

When shifting implication responsibility to GHSec:

$$\dfrac{x \notin FV(\widetilde{\phi}') \qquad x \notin FV(e) \qquad [e : T]}{\vdash \{(x : T) * [\![e]\!]_{acc} * [e : T] * \widetilde{\phi}'\} x := e \{[\![e]\!]_{acc} * \widetilde{\phi}' * (x = e)\}} \textsc{GHVarAssign}$$

Worth noting: the free variable part is only there to prevent creating false/unsatisfiable post-condition. Also, there is just no way to and point in preventing later instantiations of $\widetilde{\phi}'$ that contain x - yet at those places it is always ensured that the instantiation is still satisfiable. Could we redefine this rule (or the hoare rules in general) to simply state that deriavations are only valid on satisfiable pre- and post-conditions?

$$\dfrac{x \notin FV(e) \qquad [e : T]}{\vdash \{(x : T) * [\![e]\!]_{acc} * [e : T] * \widetilde{\phi}'\} x := e \{[\![e]\!]_{acc} * \widetilde{\phi}' * (x = e)\}} \textsc{GHVarAssign}$$

Example derivation:

$$\{(x : T) * \texttt{acc}(y.a) * \texttt{acc}(y.a.b) * \texttt{acc}(y.a.b.c) * (y : C_y) * (y.a.b \neq \texttt{null}) * \widetilde{\phi}'\}$$

$$\{(x : T) * [\![y.a.b.c]\!]_{acc} * [y.a.b.c : T] * \widetilde{\phi}'\}$$

$$x := y.a.b.c; \qquad \begin{array}{l} x \notin FV(y.a.b.c) \\ \vdash C_y.a : C_a \\ \vdash C_a.b : C_b \\ \vdash C_b.c : T \end{array}$$

$$\{[\![y.a.b.c]\!]_{acc} * \widetilde{\phi}' * (x = y.a.b.c)\}$$

$$\{\texttt{acc}(y.a) * \texttt{acc}(y.a.b) * \texttt{acc}(y.a.b.c) * \widetilde{\phi}' * (x = y.a.b.c)\}$$

5.5 Theorems

5.5.1 Soundness of α

$$\forall \overline{\phi} : \overline{\phi} \subseteq \gamma(\alpha(\overline{\phi}))$$

5.5.2 Optimality of α

$$\forall \overline{\phi}, \widetilde{\phi} : \overline{\phi} \subseteq \gamma(\widetilde{\phi}) \implies \gamma(\alpha(\overline{\phi})) \subseteq \gamma(\widetilde{\phi})$$

6 Theorems

6.1 Invariant $invariant(H, \rho, A_d, \phi)$

6.1.1 Phi valid

$$\vdash_{\texttt{sfrm}} \phi$$

6.1.2 Phi holds

$$H, \rho, A_d \vDash \phi$$

6.1.3 Types preserved

$$\begin{aligned} &\forall e, T : \phi \vdash e : T \\ \implies\ & H, \rho \vdash e : T \end{aligned}$$

6.1.4 Heap consistent

$$\begin{aligned} \forall o, C, \mu, f, T :\ & H(o) = (C, \mu) \\ \implies\ & \texttt{fieldType}(C, f) = T \\ \implies\ & H, \rho \vdash \mu(f) : T \end{aligned}$$

6.1.5 Heap not total

$$\begin{aligned} &\exists o_{min} : \\ \forall o \geq o_{min} :\ & o \notin \texttt{dom}(H) \\ & \wedge \forall f, (o, f) \notin A \end{aligned}$$

6.2 Soundness

6.2.1 Progress

$$\begin{aligned} \forall \ldots :\quad & \vdash \{\phi_1\} s' \{\phi_2\} \\ \implies\ & \mathit{invariant}(H_1, \rho_1, A_1, \phi_1) \\ \implies\ & \exists H_2, \rho_2, A_2 : (H_1, (\rho_1, A_1, s'; \overline{s}) \cdot S) \rightarrow^* (H_2, (\rho_2, A_2, \overline{s}) \cdot S) \end{aligned}$$

6.2.2 Preservation

$$\begin{aligned} \forall \ldots :\quad & \vdash \{\phi_1\} s' \{\phi_2\} \\ \implies\ & \mathit{invariant}(H_1, \rho_1, A_1, \phi_1) \\ \implies\ & (H_1, (\rho_1, A_1, s'; \overline{s}) \cdot S) \rightarrow^* (H_2, (\rho_2, A_2, \overline{s}) \cdot S) \\ \implies\ & \mathit{invariant}(H_2, \rho_2, A_2, \phi_2) \end{aligned}$$