

1 Syntax

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

2 Assumptions

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

2.0.1 Well-formed program (*program* OK)

$$\frac{\overline{cls_i \text{ OK}}}{(\overline{cls_i} \ \bar{s}) \text{ OK}} \text{ OKPROGRAM}$$

2.0.2 Well-formed class (*cls* OK)

$$\frac{\text{unique } field\text{-names} \quad \text{unique } method\text{-names} \quad \overline{method_i \text{ OK in } C}}{(\text{class } C \ \{\overline{field_i} \ \overline{method_i}\}) \text{ OK}} \text{ OKCLASS}$$

2.0.3 Well-formed method (*method* OK in *C*)

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

3 Static semantics

3.1 Expressions (*A_s* \vdash_{sfrm} *e*)

$$\frac{}{A \vdash_{\text{sfrm}} x} \text{ WFVAR}$$

$$\frac{}{A \vdash_{\text{sfrm}} v} \text{WFVALUE}$$

$$\frac{(e, f) \in A \quad A \vdash_{\text{sfrm}} e}{A \vdash_{\text{sfrm}} e.f} \text{WFFIELD}$$

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

$$\frac{}{A \vdash_{\text{sfrm}} \text{true}} \text{WFTTRUE}$$

$$\frac{A \vdash_{\text{sfrm}} e_1 \quad A \vdash_{\text{sfrm}} e_2}{A \vdash_{\text{sfrm}} (e_1 = e_2)} \text{WFEQUAL}$$

$$\frac{A \vdash_{\text{sfrm}} e_1 \quad A \vdash_{\text{sfrm}} e_2}{A \vdash_{\text{sfrm}} (e_1 \neq e_2)} \text{WFNEQUAL}$$

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

$$\frac{}{A \vdash_{\text{sfrm}} (x : T)} \text{WFTYPE}$$

$$\frac{A_s \vdash_{\text{sfrm}} \phi_1 \quad A_s \cup [\phi_1] \vdash_{\text{sfrm}} \phi_2}{A_s \vdash_{\text{sfrm}} \phi_1 * \phi_2} \text{WFSEPOP}$$

3.2.1 Implication ($\phi_1 \Rightarrow \phi_2$)

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

3.3 Footprint ($[\phi] = A_s$)

$$\begin{array}{ll} [\text{true}] & = \emptyset \\ [e_1 = e_2] & = \emptyset \\ [e_1 \neq e_2] & = \emptyset \\ [\text{acc}(e.f)] & = \{(e, f)\} \\ [\phi_1 * \phi_2] & = [\phi_1] \cup [\phi_2] \end{array}$$

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

$$\frac{}{\phi \vdash n : \text{int}} \text{STVALNUM}$$

$$\frac{}{\phi \vdash \text{null} : T} \text{STVALNULL}$$

$$\frac{\phi \Longrightarrow (x : T)}{\phi \vdash x : T} \text{STVAR}$$

$$\frac{\phi \vdash e : C \quad \vdash C.f : T}{\phi \vdash e.f : T} \text{STFIELD}$$

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

$$\frac{\vdash \{\phi_p\} s_1 \{\phi_{q1}\} \quad \phi_{q1} \Longrightarrow \phi_{q2} \quad \emptyset \vdash_{\text{sfrm}} \phi_{q2} \quad \vdash \{\phi_{q2}\} s_2 \{\phi_r\}}{\vdash \{\phi_p\} s_1; s_2 \{\phi_r\}} \text{HSEC}$$

$$\frac{\phi \Longrightarrow \phi' \quad \emptyset \vdash_{\text{sfrm}} \phi' \quad x \notin FV(\phi') \quad \phi \vdash x : C \quad \text{fields}(C) = \bar{f}}{\vdash \{\phi\} x := \text{new } C \{ \text{acc}(x, f_i) * x : C * (x \neq \text{null}) * \phi' \}} \text{HNEWOBJ}$$

$$\frac{\phi \Longrightarrow \text{acc}(x.f) * (x \neq \text{null}) * \phi' \quad \emptyset \vdash_{\text{sfrm}} \phi' \quad \phi \vdash x : C \quad \phi \vdash y : T \quad \vdash C.f : T}{\vdash \{\phi\} x.f := y \{ x : C * \text{acc}(x.f) * (x \neq \text{null}) * (x.f = y) * \phi' \}} \text{HFIELDASSIGN}$$

$$\frac{\phi \Longrightarrow \phi' \quad \emptyset \vdash_{\text{sfrm}} \phi' \quad x \notin FV(\phi') \quad x \notin FVe(e) \quad \phi \vdash x : T \quad \phi \vdash e : T \quad [\phi'] \vdash_{\text{sfrm}} e}{\vdash \{\phi\} x := e \{ \phi' * (x = e) \}} \text{HVARASSIGN}$$

$$\frac{\phi \Longrightarrow \phi' \quad \emptyset \vdash_{\text{sfrm}} \phi' \quad \text{result} \notin FV(\phi') \quad \phi \vdash x : T \quad \phi \vdash \text{result} : T}{\vdash \{\phi\} \text{return } x \{ \text{result} : T * (\text{result} = x) * \phi' \}} \text{HRETURN}$$

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

$$\frac{\phi_1 \Longrightarrow \phi_2}{\vdash \{\phi_1\} \text{assert } \phi_2 \{ \phi_1 \}} \text{HASSERT}$$

$$\frac{\phi_1 \Longrightarrow \phi_2 * \phi_r \quad \emptyset \vdash_{\text{sfrm}} \phi_r}{\vdash \{\phi_1\} \text{release } \phi_2 \{ \phi_r \}} \text{HRELEASE}$$

$$\frac{\phi \Longrightarrow \phi' \quad \emptyset \vdash_{\text{sfrm}} \phi' \quad x \notin FV(\phi')}{\vdash \{\phi\} T \ x \{ x : T * (x = \text{defaultValue}(T)) * \phi' \}} \text{HDECLARE}$$

3.5.1 Hoare revisited - pre-grad minification

$$\frac{\vdash \{\phi_p\}s_1\{\phi_{q1}\} \quad \phi_{q1} \implies \phi_{q2} \quad \emptyset \vdash_{\text{sfrm}} \phi_{q2} \quad \vdash \{\phi_{q2}\}s_2\{\phi_r\}}{\vdash \{\phi_p\}s_1; s_2\{\phi_r\}} \text{HSEC}$$

$$\frac{x \notin FV(\phi) \quad \text{fields}(C) = \bar{f}}{\vdash \{(x : C) * \phi\}x := \text{new } C\{\text{acc}(x, f_i) * (x : C) * (x \neq \text{null}) * \phi\}} \text{HNEWOBJ}$$

$$\frac{\vdash C.f : T}{\vdash \{(x : C) * (y : T) * (x \neq \text{null}) * \phi * \text{acc}(x.f)\}x.f := y\{(x : C) * \text{acc}(x.f) * (x \neq \text{null}) * (x.f = y) * \phi\}} \text{HFIELDAS}$$

$$\frac{x \notin FV(\phi) \quad x \notin FV(e) \quad [e : T]_{T'}}{\vdash \{(x : T) * \llbracket e : T \rrbracket_{T'} * \phi\}x := e\{\llbracket e : T \rrbracket_{T'} * \phi * (x = e)\}} \text{HVARASSIGN}$$

$$\frac{\text{result} \notin FV(\phi)}{\vdash \{(x : T) * (\text{result} : T) * \phi\}\text{return } x\{(\text{result} : T) * (\text{result} = x) * \phi\}} \text{HRETURN}$$

$$\frac{\text{mmethod}(C, m) = T_r \ m(T_p \ z) \ \text{requires } \phi_{pre}; \ \text{ensures } \phi_{post}; \ \{_ \} \quad x \notin FV(\phi_r) \quad x \neq y \wedge x \neq z'}{\vdash \{(x : T_r) * (y : C) * (z' : T_p) * \phi_r * (y \neq \text{null}) * \phi_{pre}[y, z' / \text{this}, z]\}x := y.m(z')\{\phi_r * \phi_{post}[y, z', x / \text{this}, z, \text{result}]\}} \text{HMMETHOD}$$

$$\frac{\phi \implies \phi'}{\vdash \{\phi\}\text{assert } \phi'\{\phi\}} \text{HASSERT}$$

$$\frac{}{\vdash \{\phi * \phi'\}\text{release } \phi'\{\phi\}} \text{HRELEASE}$$

$$\frac{x \notin FV(\phi)}{\vdash \{\phi\}T \ x\{(x : T) * (x = \text{defaultValue}(T)) * \phi\}} \text{HDECLARE}$$

4 Dynamic semantics

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

$$\frac{}{H, \rho \vdash x \Downarrow \rho(x)} \text{EEVAR}$$

$$\frac{}{H, \rho \vdash v \Downarrow v} \text{EEVALUE}$$

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

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

$$\frac{}{H, \rho, A \models \mathbf{true}} \text{EATrue}$$

$$\frac{H, \rho \vdash e_1 \Downarrow v_1 \quad H, \rho \vdash e_2 \Downarrow v_2 \quad v_1 = v_2}{H, \rho, A \models (e_1 = e_2)} \text{EAEqual}$$

$$\frac{H, \rho \vdash e_1 \Downarrow v_1 \quad H, \rho \vdash e_2 \Downarrow v_2 \quad v_1 \neq v_2}{H, \rho, A \models (e_1 \neq e_2)} \text{EANEqual}$$

$$\frac{H, \rho \vdash e \Downarrow o \quad (o, f) \in A}{H, \rho, A \models \mathbf{acc}(e.f)} \text{EAAcc}$$

$$\frac{\rho(x) = v \quad H \vdash v : T}{H, \rho, A \models (x : T)} \text{EAType}$$

$$\frac{A_1 = A \setminus A_2 \quad H, \rho, A_1 \models \phi_1 \quad H, \rho, A_2 \models \phi_2}{H, \rho, A \models \phi_1 * \phi_2} \text{EASEPOp}$$

We give a denotational semantics of formulas as $\llbracket \phi \rrbracket = \{ (H, \rho, A) \mid H, \rho, A \models \phi \}$

Note: ϕ satisfiable $\iff \llbracket \phi \rrbracket \neq \emptyset$

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

$$\phi_1 \implies \phi_2 \iff \forall H, \rho, A : H, \rho, A \models \phi_1 \implies H, \rho, A \models \phi_2$$

Drawn from def. of entailment in “A Formal Semantics for Isorecursive and Equirecursive State Abstractions”.

4.3 Footprint ($\llbracket \phi \rrbracket_{H, \rho} = A_d$)

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

4.4 Small-step ($(H, S) \rightarrow (H', S)$)

$$\frac{H, \rho \vdash x \Downarrow o \quad H, \rho \vdash y \Downarrow v_y \quad (o, f) \in A \quad H' = H[o \mapsto [f \mapsto v_y]]}{(H, (\rho, A, x.f := y; \bar{s}) \cdot S) \rightarrow (H', (\rho, A, \bar{s}) \cdot S)} \text{ESFieldAssign}$$

$$\frac{H, \rho \vdash e \Downarrow v \quad \rho' = \rho[x \mapsto v]}{(H, (\rho, A, x := e; \bar{s}) \cdot S) \rightarrow (H, (\rho', A, \bar{s}) \cdot S)} \text{ESVarAssign}$$

$$\frac{\text{fields}(C) = \bar{T} \ \bar{f} \quad \rho' = \rho[x \mapsto o] \quad A' = A * (\bar{o}, \bar{f}_i) \quad H' = H[o \mapsto \overline{[f \mapsto \text{defaultValue}(\bar{T})]}]}{(H, (\rho, A, x := \mathbf{new} \ C; \bar{s}) \cdot S) \rightarrow (H', (\rho', A', \bar{s}) \cdot S)} \text{ESNewObj}$$

$$\frac{H, \rho \vdash x \Downarrow v_x \quad \rho' = \rho[\mathbf{result} \mapsto v_x]}{(H, (\rho, A, \mathbf{return} \ x; \bar{s}) \cdot S) \rightarrow (H, (\rho', A, \bar{s}) \cdot S)} \text{ESRETURN}$$

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

$$\frac{\begin{array}{c} H(o) = (C, _) \quad \text{mpost}(C, m) = \phi \quad H, \rho \vdash y \Downarrow o \\ H, \rho', A' \models \phi \quad A'' = \lfloor \phi \rfloor_{H, \rho'} \quad H, \rho' \vdash \mathbf{result} \Downarrow v_r \end{array}}{(H, (\rho', A', \emptyset) \cdot (\rho, A, x := y.m(z); \bar{s}) \cdot S) \rightarrow (H, (\rho[x \mapsto v_r], A * A'', \bar{s}) \cdot S))} \text{ESAPPFINISH}$$

$$\frac{H, \rho, A \models \phi}{(H, (\rho, A, \mathbf{assert} \ \phi; \bar{s}) \cdot S) \rightarrow (H, (\rho, A, \bar{s}) \cdot S)} \text{ESASSERT}$$

$$\frac{H, \rho, A \models \phi \quad A' = A \setminus \lfloor \phi \rfloor_{H, \rho}}{(H, (\rho, A, \mathbf{release} \ \phi; \bar{s}) \cdot S) \rightarrow (H, (\rho, A', \bar{s}) \cdot S)} \text{ESRELEASE}$$

$$\frac{\rho' = \rho[x \mapsto \mathbf{defaultValue}(T)]}{(H, (\rho, A, T \ x; \bar{s}) \cdot S) \rightarrow (H, (\rho', A, \bar{s}) \cdot S)} \text{ESDECLARE}$$

5 Gradualization

5.1 Syntax

5.1.1 Gradual formula

$$\tilde{\phi} ::= \phi \mid ? * \phi$$

Note: consider $?$ in other positions as “self-framing delimiter”, but with semantically identical meaning. As long as $?$ is only legal in the front though: $\phi_1 * \tilde{\phi}_2$ propagates the $?$ to the very left in case $\tilde{\phi}_2$ contains one.

5.1.2 Type judgment expansion

Motivation: Materialize and combine the requirements on ϕ .

Designed so that

$$\begin{aligned} \phi \vdash e : T \quad \wedge \quad \lfloor \phi \rfloor \vdash_{\mathbf{sfrm}} e \\ \iff \\ [e : T]_C \quad \wedge \quad \phi \implies \llbracket e : T \rrbracket_C \end{aligned}$$

Expand into premise: $[e : T]_C$

$$\begin{aligned} [v : T]_C &= T = C \quad \wedge \quad \vdash v : T \\ [x : T]_C &= T = C \\ [e.f : T]_C &= [e : C']_C \quad \wedge \quad \vdash C'.f : T \end{aligned}$$

Expand into formula: $\llbracket e : T \rrbracket_C$

$$\begin{aligned} \llbracket v : T \rrbracket_C &= \mathbf{true} \\ \llbracket x : T \rrbracket_C &= (x : C) \\ \llbracket e.f : T \rrbracket_C &= \llbracket e : T \rrbracket_C * \mathbf{acc}(e.f) \end{aligned}$$

5.2 Concretization

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

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

5.3 Abstraction

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

5.4 Gradual Lifting

5.4.1 Self framing

$$\frac{A \vdash_{\text{sfrm}} \phi}{A \widetilde{\vdash}_{\text{sfrm}} \phi} \text{ GSFRMNONGRAD}$$

$$\frac{}{A \widetilde{\vdash}_{\text{sfrm}} ? * \phi} \text{ GSFRMGRAD}$$

5.4.2 Implication

$$\frac{\phi_1 \implies \phi_2}{\phi_1 \widetilde{\implies} \phi_2} \text{ GIMPLNONGRAD}$$

$$\frac{\hat{\phi}_m \implies \phi_2 \quad \hat{\phi}_m \implies \phi_1}{? * \phi_1 \widetilde{\implies} \phi_2} \text{ GIMPLGRAD}$$

$\hat{\phi}_m$ 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 Equality

$$\frac{\phi_1 = \phi_2}{\phi_1 \approx \phi_2} \text{ GEQSTATIC}$$

$$\frac{\begin{array}{c} \text{at least one of } \widetilde{\phi}_1 \text{ or } \widetilde{\phi}_2 \text{ contains ?} \\ \widetilde{\phi}_1 \widetilde{\implies} \widetilde{\phi}_2 \quad \widetilde{\phi}_2 \widetilde{\implies} \widetilde{\phi}_1 \end{array}}{\widetilde{\phi}_1 \approx \widetilde{\phi}_2} \text{ GEQGRADUAL}$$

5.4.4 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:

$$\frac{\emptyset \vdash_{\text{sfrm}} \tilde{\phi}' \quad x \notin FV(\tilde{\phi}') \quad x \notin FV(e) \quad \epsilon \vdash \tilde{\phi} \widetilde{\Rightarrow} \tilde{\phi}' \quad \epsilon \vdash \tilde{\phi} \vdash x : T \quad \epsilon \vdash \tilde{\phi} \vdash e : T \quad \epsilon \vdash [\tilde{\phi}'] \vdash_{\text{sfrm}} e}{\vdash \{\tilde{\phi}\}x := e\{\tilde{\phi}' * (x = e)\}} \text{GHVARASSIGN}$$

Collapsing (hidden) gradual implications into a single one:

$$\frac{\epsilon \vdash \tilde{\phi} \widetilde{\Rightarrow} (x : T) * \llbracket e : T \rrbracket_C * \tilde{\phi}' \quad \emptyset \vdash_{\text{sfrm}} \llbracket e : T \rrbracket_C * \tilde{\phi}' \quad x \notin FV(\tilde{\phi}') \quad x \notin FV(e) \quad [e : T]_C}{\vdash \{\tilde{\phi}\}x := e\{\llbracket e : T \rrbracket_C * \tilde{\phi}' * (x = e)\}} \text{GHVARASSIGN}$$

When shifting implication responsibility to GHSec:

$$\frac{x \notin FV(\tilde{\phi}') \quad x \notin FV(e) \quad [e : T]_C}{\vdash \{(x : T) * \llbracket e : T \rrbracket_C * \tilde{\phi}'\}x := e\{\llbracket e : T \rrbracket_C * \tilde{\phi}' * (x = e)\}} \text{GHVARASSIGN}$$

Example derivation:

$$\begin{aligned} & \{(x : T) * (y : C) * \text{acc}(y.a) * \text{acc}(y.a.b) * \text{acc}(y.a.b.c) * \tilde{\phi}'\} \\ & \{(x : T) * \llbracket y.a.b.c : T \rrbracket_C * \tilde{\phi}'\} \\ & x := y.a.b.c; \quad \begin{array}{l} x \notin FV(\tilde{\phi}') \\ x \notin FV(y.a.b.c) \\ [y.a.b.c : T]_C = \vdash C_y = C \wedge \vdash C_y.a : C_a \wedge \vdash C_a.b : C_b \wedge \vdash C_b.c : T \end{array} \\ & \{\llbracket y.a.b.c : T \rrbracket_C * \tilde{\phi}' * (x = y.a.b.c)\} \\ & \{(y : C) * \text{acc}(y.a) * \text{acc}(y.a.b) * \text{acc}(y.a.b.c) * \tilde{\phi}' * (x = y.a.b.c)\} \end{aligned}$$

5.4.5 GHFieldAssign

$$\frac{\vdash_{\text{sfrm}} \phi \quad \vdash C.f : T \quad \tilde{\phi}_1 \approx (x : C) * (y : T) * (x \neq \text{null}) * \phi * \text{acc}(x.f) \quad \tilde{\phi}_2 \approx (x : C) * \text{acc}(x.f) * (x \neq \text{null}) * (x.f = y) * \phi}{\vdash \{\tilde{\phi}_1\}x.f := y\{\tilde{\phi}_2\}} \text{GHFIELDASSIGN}$$

5.4.6 GHSec - sound but obviously not complete!

$$\frac{\tilde{\vdash}\{\tilde{\phi}_p\}s_1\{\tilde{\phi}_{q1}\} \quad \phi_{q1} \Longrightarrow \phi_{q2} \quad \emptyset \vdash_{\text{sfrm}} \phi_{q2} \quad \tilde{\vdash}\{\phi_{q2}\}s_2\{\tilde{\phi}_r\}}{\tilde{\vdash}\{\tilde{\phi}_p\}s_1; s_2\{\tilde{\phi}_r\}} \text{HSEC}$$

5.5 Theorems

5.5.1 Soundness of α

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

5.5.2 Optimality of α

$$\forall \bar{\phi}, \tilde{\phi} : \bar{\phi} \subseteq \gamma(\tilde{\phi}) \Longrightarrow \gamma(\alpha(\bar{\phi})) \subseteq \gamma(\tilde{\phi})$$

6 Theorems

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

6.1.1 Phi valid

$$\vdash_{\text{sfrm}} \phi$$

6.1.2 Phi holds

$$H, \rho, A_d \models \phi$$

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

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

6.1.5 Heap not total

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

6.2 Soundness

6.2.1 Progress

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

6.2.2 Preservation

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