1 Syntax

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

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* OK)

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

2.0.2 Well-formed class (*cls* OK)

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

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

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

3 Static semantics

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

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

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

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

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

$$\frac{}{A \vdash_{\texttt{sfrm}} \texttt{true}} \text{ WFTRUE}$$

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

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

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

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

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

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

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 ($\phi \vdash e : T$)

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

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

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

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

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

$$\frac{\vdash \{\phi_p\}s_1\{\phi_{q1}\} \qquad \phi_{q1} \implies \phi_{q2} \qquad \emptyset \vdash_{\texttt{sfrm}} \phi_{q2} \qquad \vdash \{\phi_{q2}\}s_2\{\phi_r\}}{\vdash \{\phi_p\}s_1; s_2\{\phi_r\}} \text{ HSec}$$

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

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

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

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

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

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

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

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

3.5.1   Hoare revisited - pre-grad minification

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

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

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

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

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

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

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

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

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

3.5.2   Hoare revisited - pre-grad minification with simple HSec

$$\frac{\vdash \{\phi_p\}s_1\{\phi_q\} \qquad \vdash \{\phi_q\}s_2\{\phi_r\}}{\vdash \{\phi_p\}s_1; s_2\{\phi_r\}} \text{ HSEC}$$

(other rules not printed here: think of previous subsection, but with self-framed implication on every pre-condition)

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

$$\frac{}{H, \rho, A \vDash \mathtt{true}} \ \text{EATRUE}$$

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

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

$$\frac{H, \rho \vdash e \Downarrow o \qquad (o, f) \in A}{H, \rho, A \vDash \mathtt{acc}(e.f)} \ \text{EAACC}$$

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\mathtt{fields}(C) = \overline{T} \ \overline{f} \qquad \rho' = \rho[x \mapsto o] \qquad A' = A * \overline{(o, f_i)} \qquad H' = H[o \mapsto [\overline{f \mapsto \mathtt{defaultValue}(T)}]]}{(H, (\rho, A, x := \mathtt{new} \ C; \overline{s}) \cdot S) \to (H', (\rho', A', \overline{s}) \cdot S)} \ \text{ESNEWOBJ}$$

with condition $o \notin \mathtt{dom}(H)$

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

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

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

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

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

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

5 Gradualization

5.1 Syntax

5.1.1 Gradual formula

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

Note: consider ? in other positions as "self-framing delimiter", but with semantically identical meaning.
As long as ? is only legal in the front though: $\phi_1 * \widetilde{\phi_2}$ propagates the ? to the very left in case $\widetilde{\phi_2}$ contains one.

5.1.2 Type judgment expansion

Motivation: Materialize and combine the requirements on ϕ.
Designed so that

$$\phi \vdash e : T \quad \wedge \quad \lfloor \phi \rfloor \vdash_{\texttt{sfrm}} e$$

$$\Longleftrightarrow$$

$$[e : T]_C \quad \wedge \quad \phi \implies [\![e : T]\!]_C$$

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

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

Expand into formula: $[\![e : T]\!]_C$

$$\begin{aligned} [\![v : T]\!]_C &= \texttt{true} \\ [\![x : T]\!]_C &= (x : C) \\ [\![e.f : T]\!]_C &= [\![e : T]\!]_C * \texttt{acc}(e.f) \end{aligned}$$

5.2 Concretization

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

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

5.3 Abstraction

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

5.4 Gradual Lifting

5.4.1 Self framing

$$\frac{A \vdash_{\mathtt{sfrm}} \phi}{A \mathrel{\widetilde{\vdash_{\mathtt{sfrm}}}} \phi}\ \text{GSfrmNonGrad}$$

$$\frac{}{A \mathrel{\widetilde{\vdash_{\mathtt{sfrm}}}} ?\ *\ \phi}\ \text{GSfrmGrad}$$

5.4.2 Implication

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

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

$\hat{\phi_m}$ 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

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

$$\frac{\text{at least one of } \widetilde{\phi_1} \text{ or } \widetilde{\phi_2} \text{ contains } ?\qquad \widetilde{\phi_1} \mathrel{\widetilde{\implies}} \widetilde{\phi_2} \qquad \widetilde{\phi_2} \mathrel{\widetilde{\implies}} \widetilde{\phi_1}}{\widetilde{\phi_1} \approx \widetilde{\phi_2}}\ \text{GEqGradual}$$

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:

$$\frac{\emptyset \vdash_{\texttt{sfrm}} \widetilde{\phi'} \qquad x \notin FV(\widetilde{\phi'}) \qquad x \notin FV(e) \qquad \overset{\epsilon \vdash \widetilde{\phi} \overset{\Longrightarrow}{\Longrightarrow} \widetilde{\phi'}}{\epsilon \vdash \widetilde{\phi} \vdash x : T} \qquad \epsilon \vdash \widetilde{\phi} \vdash e : T \qquad \epsilon \vdash \lfloor \widetilde{\phi'} \rfloor \vdash_{\texttt{sfrm}} e}{\vdash \{\widetilde{\phi}\} x := e \{\widetilde{\phi'} * (x = e)\}} \text{ GHVARAssign}$$

Collapsing (hidden) gradual implications into a single one:

$$\frac{\epsilon \vdash \widetilde{\phi} \overset{\Longrightarrow}{\Longrightarrow} (x : T) * [\![e : T]\!]_C * \widetilde{\phi'} \qquad \emptyset \vdash_{\texttt{sfrm}} [\![e : T]\!]_C * \widetilde{\phi'} \qquad x \notin FV(\widetilde{\phi'}) \qquad x \notin FV(e) \qquad [e : T]_C}{\vdash \{\widetilde{\phi}\} x := e \{[\![e : T]\!]_C * \widetilde{\phi'} * (x = e)\}} \text{ GHVARAssig}$$

When shifting implication responsibility to GHSec:

$$\frac{x \notin FV(\widetilde{\phi'}) \qquad x \notin FV(e) \qquad [e : T]_C}{\vdash \{(x : T) * [\![e : T]\!]_C * \widetilde{\phi'}\} x := e \{[\![e : T]\!]_C * \widetilde{\phi'} * (x = e)\}} \text{ GHVARAssign}$$

Example derivation:

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

$$x := y.a.b.c; \qquad \begin{array}{l} x \notin FV(\widetilde{\phi'}) \\ x \notin FV(y.a.b.c) \\ [y.a.b.c : T]_C = \ \vdash C_y = C \ \wedge \ \vdash C_y.a : C_a \ \wedge \ \vdash C_a.b : C_b \ \wedge \ \vdash C_b.c : T \end{array}$$

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

5.4.5 GHFieldAssign

$$\frac{\widetilde{\phi_1} \approx (x : C) * (y : T) * (x \neq \texttt{null}) * \phi * \texttt{acc}(x.f) \qquad \overset{\vdash_{\texttt{sfrm}} \phi \qquad \vdash C.f : T}{\widetilde{\phi_2} \approx (x : C) * \texttt{acc}(x.f) * (x \neq \texttt{null}) * (x.f = y) * \phi}}{\widetilde{\vdash}\{\widetilde{\phi_1}\} x.f := y \{\widetilde{\phi_2}\}} \text{ GHFIELDA}$$

5.4.6 GHSec - sound but obviously not complete!

$$\frac{\widetilde{\vdash}\{\widetilde{\phi_p}\} s_1 \{\widetilde{\phi_{q1}}\} \qquad \phi_{q1} \Longrightarrow \phi_{q2} \qquad \emptyset \vdash_{\texttt{sfrm}} \phi_{q2} \qquad \widetilde{\vdash}\{\phi_{q2}\} s_2 \{\widetilde{\phi_r}\}}{\widetilde{\vdash}\{\widetilde{\phi_p}\} s_1; s_2 \{\widetilde{\phi_r}\}} \text{ GHSEC}$$

5.4.7 GHFieldAssign (implication alternative)

$$\frac{\phi_1 \Longrightarrow (x : C) * (y : T) * (x \neq \texttt{null}) * \phi * \texttt{acc}(x.f) \qquad \overset{\vdash_{\texttt{sfrm}} \phi \qquad \vdash C.f : T}{\phi_2 = (x : C) * \texttt{acc}(x.f) * (x \neq \texttt{null}) * (x.f = y) * \phi}}{\vdash \{\phi_1\} x.f := y \{\phi_2\}} \text{ HFIELDA}$$

$$\frac{\widetilde{\phi_1} \overset{\Longrightarrow}{\Longrightarrow} (x : C) * (y : T) * (x \neq \texttt{null}) * \phi * \texttt{acc}(x.f) \qquad \overset{\vdash_{\texttt{sfrm}} \phi \qquad \vdash C.f : T}{\widetilde{\phi_2} \approx (x : C) * \texttt{acc}(x.f) * (x \neq \texttt{null}) * (x.f = y) * \phi}}{\widetilde{\vdash}\{\widetilde{\phi_1}\} x.f := y \{\widetilde{\phi_2}\}} \text{ GHFIELD}$$

Note: With this alternative rule design $\widetilde{\Longrightarrow}$ is consistently used with static formulas as second argument. This plays nicely with the fact that $\widetilde{\Longrightarrow}$ does not care about the gradualness of that argument. Might make sense to define lifting of $\Longrightarrow$ as lifting on only the first parameter in the first place.

Minimum runtime checks: For $\widetilde{\phi_1} \widetilde{\Longrightarrow} \widetilde{\phi_2}$ to hold at runtime, practically just ϕ_2 needs to hold, so that would be a valid assertion to check. Yet, we know statically that ϕ_1 holds, so we can remove everything from the runtime check that is implied by ϕ_1. So in a sense, we only need to check $\phi_2 \backslash \phi_1$ at runtime (the operator can be an approximation).

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

$$\forall e, T : \phi \vdash e : T$$
$$\implies H, \rho \vdash e : T$$

6.1.4 Heap consistent

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

6.1.5 Heap not total

$$\exists o_{min} :$$
$$\forall o \geq o_{min} : o \notin \texttt{dom}(H)$$
$$\land \forall f, (o, f) \notin A$$

6.2 Soundness

6.2.1 Progress

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

6.2.2   Preservation

$$\forall \dots : \quad \vdash \{\phi_1\} s' \{\phi_2\}$$
$$\implies invariant(H_1, \rho_1, A_1, \phi_1)$$
$$\implies (H_1, (\rho_1, A_1, s'; \overline{s}) \cdot S) \to^* (H_2, (\rho_2, A_2, \overline{s}) \cdot S)$$
$$\implies invariant(H_2, \rho_2, A_2, \phi_2)$$

$$\forall \dots : \quad \vdash \{\phi_1\} s' \{\phi_2\}$$
$$\implies invariant(H_1, \rho_1, A_1, \phi_1)$$
$$\implies (H_1, (\rho_1, A_1, s'; \overline{s}) \cdot S) \to^* (H_2, (\rho_2, A_2, \overline{s}) \cdot S)$$
$$\implies invariant(H_2, \rho_2, A_2, \phi_2)$$