

# 1 Syntax

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

## 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*)

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

## 3 Static semantics

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

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

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

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

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

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

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

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

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

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

#### 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 ( $\Gamma \vdash e : T$ )

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

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

$$\frac{\Gamma(x) = T}{\Gamma \vdash x : T} \text{STVAR}$$

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

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

$$\begin{array}{c}
\frac{\phi \implies \phi' \quad \emptyset \vdash_{\text{sfrm}} \phi' \quad x \notin FV(\phi') \quad \Gamma \vdash x : C \quad \text{fields}(C) = \bar{f}}{\Gamma \vdash \{\phi\} x := \text{new } C \{\phi' * (x \neq \text{null}) * \text{acc}(x, f_i)\}} \text{HNEWOBJ} \\[2ex]
\frac{\phi \implies \text{acc}(x.f) * (x \neq \text{null}) * \phi' \quad \emptyset \vdash_{\text{sfrm}} \phi' \quad \Gamma \vdash x : C \quad \Gamma \vdash y : T \quad \vdash C.f : T}{\Gamma \vdash \{\phi\} x.f := y \{\phi' * \text{acc}(x.f) * (x \neq \text{null}) * (x.f = y)\}} \text{HFIELDASSIGN} \\[2ex]
\frac{\phi \implies \phi' \quad \emptyset \vdash_{\text{sfrm}} \phi' \quad x \notin FV(\phi') \quad x \notin FV(e) \quad \Gamma \vdash x : T \quad \Gamma \vdash e : T \quad \llbracket e \rrbracket \subseteq \phi'}{\Gamma \vdash \{\phi\} x := e \{\phi' * (x = e)\}} \text{HVARASSIGN} \\[2ex]
\frac{\phi \implies \phi' \quad \emptyset \vdash_{\text{sfrm}} \phi' \quad \text{result} \notin FV(\phi') \quad \Gamma \vdash x : T \quad \Gamma \vdash \text{result} : T}{\Gamma \vdash \{\phi\} \text{return } x \{\phi' * (\text{result} = x)\}} \text{HRETURN} \\[2ex]
\frac{\begin{array}{c} \Gamma \vdash y : C \quad \text{mmethod}(C, m) = T_r \ m(T_p \ z) \text{ requires } \phi_{pre}; \text{ ensures } \phi_{post}; \{\_\} \\ \Gamma \vdash x : T_r \quad \Gamma \vdash z' : T_p \quad \phi \implies (y \neq \text{null}) * \phi_p * \phi' \quad \emptyset \vdash_{\text{sfrm}} \phi' \\ x \notin FV(\phi') \quad x \neq y \wedge x \neq z' \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}}{\Gamma \vdash \{\phi\} x := y.m(z') \{\phi' * \phi_q\}} \text{HAPP} \\[2ex]
\frac{\phi \implies \phi'}{\Gamma \vdash \{\phi\} \text{assert } \phi' \{\phi\}} \text{HASSERT} \\[2ex]
\frac{\phi \implies \phi_r * \phi' \quad \emptyset \vdash_{\text{sfrm}} \phi'}{\Gamma \vdash \{\phi\} \text{release } \phi_r \{\phi'\}} \text{HRELEASE} \\[2ex]
\frac{x \notin \text{dom}(\Gamma) \quad \Gamma, x : T \vdash \{(x = \text{defaultValue}(T)) * \phi\} \bar{s}\{\phi'\}}{\Gamma \vdash \{\phi\} T \ x; \bar{s}\{\phi'\}} \text{HDECLARE} \\[2ex]
\frac{\Gamma \vdash \{\phi_p\} s_1 \{\phi_q\} \quad \Gamma \vdash \{\phi_q\} s_2 \{\phi_r\}}{\Gamma \vdash \{\phi_p\} s_1; s_2 \{\phi_r\}} \text{HSEC}
\end{array}$$

#### 3.5.1 Notation

$$\begin{array}{c}
\frac{\hat{\phi} \implies \hat{\phi}' \quad x \notin FV(\hat{\phi}') \quad \Gamma \vdash x : C \quad \text{fields}(C) = \bar{f}}{\Gamma \vdash \{\hat{\phi}\} x := \text{new } C \{\hat{\phi}' * (x \neq \text{null}) * \text{acc}(x, f_i)\}} \text{HNEWOBJ} \\[2ex]
\frac{\hat{\phi} \implies \text{acc}(x.f) * (x \neq \text{null}) * \hat{\phi}' \quad \Gamma \vdash x : C \quad \Gamma \vdash y : T \quad \vdash C.f : T}{\Gamma \vdash \{\hat{\phi}\} x.f := y \{\hat{\phi}' * \text{acc}(x.f) * (x \neq \text{null}) * (x.f = y)\}} \text{HFIELDASSIGN} \\[2ex]
\frac{\hat{\phi} \implies \hat{\phi}' \quad x \notin FV(\hat{\phi}') \quad x \notin FV(e) \quad \Gamma \vdash x : T \quad \Gamma \vdash e : T \quad \llbracket e \rrbracket \subseteq \hat{\phi}'}{\Gamma \vdash \{\hat{\phi}\} x := e \{\hat{\phi}' * (x = e)\}} \text{HVARASSIGN}
\end{array}$$

$$\frac{\hat{\phi} \Longrightarrow \hat{\phi}' \quad \mathbf{result} \notin FV(\hat{\phi}') \quad \Gamma \vdash x : T \quad \Gamma \vdash \mathbf{result} : T}{\Gamma \vdash \{\hat{\phi}\} \mathbf{return} x \{\hat{\phi}' * (\mathbf{result} = x)\}} \text{HRETURN}$$

$$\frac{\Gamma \vdash y : C \quad \mathbf{mmethod}(C, m) = T_r \ m(T_p \ z) \ \mathbf{requires} \ \hat{\phi}_{pre}; \ \mathbf{ensures} \ \hat{\phi}_{post}; \ \{\_\} \quad \Gamma \vdash x : T_r \quad \Gamma \vdash z' : T_p \quad \hat{\phi} \Longrightarrow (y \neq \mathbf{null}) * \hat{\phi}_p * \hat{\phi}' \quad x \notin FV(\hat{\phi}') \quad x \neq y \wedge x \neq z' \quad \hat{\phi}_p = \hat{\phi}_{pre}[y, z' / \mathbf{this}, z] \quad \hat{\phi}_q = \hat{\phi}_{post}[y, z', x / \mathbf{this}, z, \mathbf{result}]}{\Gamma \vdash \{\hat{\phi}\} x := y.m(z') \{\hat{\phi}' * \hat{\phi}_q\}} \text{HAPP}$$

$$\frac{\hat{\phi} \Longrightarrow \phi'}{\Gamma \vdash \{\hat{\phi}\} \mathbf{assert} \ \phi' \{\hat{\phi}\}} \text{HASSERT}$$

$$\frac{\hat{\phi} \Longrightarrow \phi_r * \hat{\phi}'}{\Gamma \vdash \{\hat{\phi}\} \mathbf{release} \ \phi_r \{\hat{\phi}'\}} \text{HRELEASE}$$

$$\frac{x \notin \mathbf{dom}(\Gamma) \quad \Gamma, x : T \vdash \{\hat{\phi} * (x = \mathbf{defaultValue}(T))\} \bar{s} \{\hat{\phi}'\}}{\Gamma \vdash \{\hat{\phi}\} T \ x; \bar{s} \{\hat{\phi}'\}} \text{HDECLARE}$$

$$\frac{\Gamma \vdash \{\hat{\phi}_p\} s_1 \{\hat{\phi}_q\} \quad \Gamma \vdash \{\hat{\phi}_q\} s_2 \{\hat{\phi}_r\}}{\Gamma \vdash \{\hat{\phi}_p\} s_1; s_2 \{\hat{\phi}_r\}} \text{HSEC}$$

### 3.5.2 Deterministic

$$\frac{\hat{\phi}[\mathbf{w/o} \ x] = \hat{\phi}' \quad \Gamma \vdash x : C \quad \mathbf{fields}(C) = \bar{f}}{\Gamma \vdash \{\hat{\phi}\} x := \mathbf{new} \ C \{\hat{\phi}' * (x \neq \mathbf{null}) * \mathbf{acc}(x, f_i)\}} \text{HNEWOBJ}$$

$$\frac{\hat{\phi}[\mathbf{w/o} \ \mathbf{acc}(x.f)] = \hat{\phi}' \quad \hat{\phi} \Longrightarrow \mathbf{acc}(x.f) * (x \neq \mathbf{null}) \quad \Gamma \vdash x : C \quad \Gamma \vdash y : T \quad \vdash C.f : T}{\Gamma \vdash \{\hat{\phi}\} x.f := y \{\hat{\phi}' * \mathbf{acc}(x.f) * (x \neq \mathbf{null}) * (x.f = y)\}} \text{HFIELDASSIGN}$$

$$\frac{\hat{\phi}[\mathbf{w/o} \ x] = \hat{\phi}' \quad x \notin FV(e) \quad \Gamma \vdash x : T \quad \Gamma \vdash e : T \quad \llbracket e \rrbracket \subseteq \hat{\phi}'}{\Gamma \vdash \{\hat{\phi}\} x := e \{\hat{\phi}' * (x = e)\}} \text{HVARASSIGN}$$

$$\frac{\hat{\phi}[\mathbf{w/o} \ \mathbf{result}] = \hat{\phi}' \quad \Gamma \vdash x : T \quad \Gamma \vdash \mathbf{result} : T}{\Gamma \vdash \{\hat{\phi}\} \mathbf{return} x \{\hat{\phi}' * (\mathbf{result} = x)\}} \text{HRETURN}$$

$$\frac{\Gamma \vdash y : C \quad \mathbf{mmethod}(C, m) = T_r \ m(T_p \ z) \ \mathbf{requires} \ \hat{\phi}_{pre}; \ \mathbf{ensures} \ \hat{\phi}_{post}; \ \{\_\} \quad \hat{\phi} \Longrightarrow (y \neq \mathbf{null}) * \hat{\phi}_p \quad x \neq y \wedge x \neq z' \quad \hat{\phi}_p = \hat{\phi}_{pre}[y, z' / \mathbf{this}, z] \quad \hat{\phi}_q = \hat{\phi}_{post}[y, z', x / \mathbf{this}, z, \mathbf{result}]}{\Gamma \vdash \{\hat{\phi}\} x := y.m(z') \{\hat{\phi}' * \hat{\phi}_q\}} \text{HAPP}$$

$$\frac{\hat{\phi} \Longrightarrow \phi'}{\Gamma \vdash \{\hat{\phi}\} \mathbf{assert} \ \phi' \{\hat{\phi}\}} \text{HASSERT}$$

$$\frac{\hat{\phi}[\mathbf{w}/\mathbf{o} \mid \phi_r] = \hat{\phi}' \quad \hat{\phi} \Longrightarrow \phi_r}{\Gamma \vdash \{\hat{\phi}\} \mathbf{release} \phi_r \{\hat{\phi}'\}} \text{HRELEASE}$$

$$\frac{x \notin \text{dom}(\Gamma) \quad \Gamma, x : T \vdash \{\hat{\phi} \hat{*} (x = \text{defaultValue}(T))\} \bar{s}\{\hat{\phi}'\}}{\Gamma \vdash \{\hat{\phi}\} T x; \bar{s}\{\hat{\phi}'\}} \text{HDECLARE}$$

$$\frac{\Gamma \vdash \{\hat{\phi}_p\} s_1 \{\hat{\phi}_q\} \quad \Gamma \vdash \{\hat{\phi}_q\} s_2 \{\hat{\phi}_r\}}{\Gamma \vdash \{\hat{\phi}_p\} s_1; s_2 \{\hat{\phi}_r\}} \text{HSEC}$$

### 3.5.3 HApp determinism discussion

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

$$\frac{\begin{array}{l} \Gamma \vdash y : C \quad \text{mmethod}(C, m) = T_r \ m(T_p \ z) \ \mathbf{requires} \ \phi_{pre}; \ \mathbf{ensures} \ \phi_{post}; \ \{\_\} \\ \Gamma \vdash x : T_r \quad \Gamma \vdash z' : T_p \quad \phi \Longrightarrow (y \neq \mathbf{null}) * \phi_p \\ \phi_r = \phi / \phi_p / x \quad x \neq y \wedge x \neq z' \quad \phi_p = \phi_{pre}[y, z' / \mathbf{this}, z] \quad \phi_q = \phi_{post}[y, z', x / \mathbf{this}, z, \mathbf{result}] \end{array}}{\Gamma \vdash \{\phi\} x := y.m(z') \{\phi_q * \phi_r\}} \text{HAPPD}$$

$$\frac{\begin{array}{l} \Gamma \vdash y : C \quad \text{mmethod}(C, m) = T_r \ m(T_p \ z) \ \mathbf{requires} \ \widetilde{\phi_{pre}}; \ \mathbf{ensures} \ \widetilde{\phi_{post}}; \ \{\_\} \\ \Gamma \vdash x : T_r \quad \Gamma \vdash z' : T_p \quad \widetilde{\phi} \Longrightarrow (y \neq \mathbf{null}) * \phi_p \\ \widetilde{\phi}_r = \widetilde{\phi} / y : C, z' : T_p, x : T_r \widetilde{\phi}_p / x \quad x \neq y \wedge x \neq z' \quad \widetilde{\phi}_p = \widetilde{\phi_{pre}}[y, z' / \mathbf{this}, z] \quad \widetilde{\phi}_q = \widetilde{\phi_{post}}[y, z', x / \mathbf{this}, z, \mathbf{result}] \end{array}}{\Gamma \vdash \{\widetilde{\phi}\} x := y.m(z') \{\widetilde{\phi}_q * \widetilde{\phi}_r\}} \text{GHAPP}$$

## 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 ( $H, \rho, A \models \phi$ )

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

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

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

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

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

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

Note:  $\phi$  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”.

#### 4.2.2 Implying inequality

$$\begin{aligned} & \phi * (e_1 = e_1) * (e_2 = e_2) \implies (e_1 \neq e_2) \\ = & \forall H, \rho, A : H, \rho, A \models \phi * (e_1 = e_1) * (e_2 = e_2) \implies H, \rho, A \models (e_1 \neq e_2) \\ = & \forall H, \rho, A : (\exists v_1, v_2 : H, \rho \vdash e_1 \Downarrow v_1 \wedge H, \rho \vdash e_2 \Downarrow v_2 \wedge H, \rho, A \models \phi) \implies (\exists v_1, v_2 : H, \rho \vdash e_1 \Downarrow v_1 \wedge H, \rho \vdash e_2 \Downarrow v_2 \wedge v_1 \neq v_2) \\ = & \forall H, \rho, A, v_1, v_2 : (H, \rho \vdash e_1 \Downarrow v_1 \wedge H, \rho \vdash e_2 \Downarrow v_2 \wedge H, \rho, A \models \phi) \implies (\exists v_1, v_2 : H, \rho \vdash e_1 \Downarrow v_1 \wedge H, \rho \vdash e_2 \Downarrow v_2 \wedge v_1 \neq v_2) \\ = & \forall H, \rho, A, v_1, v_2 : (H, \rho \vdash e_1 \Downarrow v_1 \wedge H, \rho \vdash e_2 \Downarrow v_2 \wedge H, \rho, A \models \phi) \implies (v_1 \neq v_2) \\ = & \forall H, \rho, A, v_1, v_2 : \neg(H, \rho \vdash e_1 \Downarrow v_1 \wedge H, \rho \vdash e_2 \Downarrow v_2 \wedge H, \rho, A \models \phi) \vee (v_1 \neq v_2) \\ = & \forall H, \rho, A, v_1, v_2 : \neg(H, \rho \vdash e_1 \Downarrow v_1 \wedge H, \rho \vdash e_2 \Downarrow v_2 \wedge H, \rho, A \models \phi \wedge (v_1 = v_2)) \\ = & \forall H, \rho, A : \neg(\exists v_1, v_2 : H, \rho \vdash e_1 \Downarrow v_1 \wedge H, \rho \vdash e_2 \Downarrow v_2 \wedge H, \rho, A \models \phi \wedge (v_1 = v_2)) \\ = & \forall H, \rho, A : \neg(H, \rho, A \models \phi \wedge H, \rho, A \models (e_1 = e_2)) \\ = & \forall H, \rho, A : \neg H, \rho, A \models \phi * (e_1 = e_2) \\ = & \neg \text{sat} (\phi * (e_1 = e_2)) \end{aligned}$$

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

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

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

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

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

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

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

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

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

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

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

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

## 5 Gradualization

### 5.1 Syntax

#### 5.1.1 Gradual formula

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

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

#### 5.1.2 Self-framed and satisfiable formula

$$\hat{\phi} \in \{ \phi \mid \vdash_{\text{sfrm}} \phi \wedge \text{sat } \phi \}$$

### 5.2 Concretization

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

$$\widetilde{\phi_1} \sqsubseteq \widetilde{\phi_2} \quad : \iff \quad \gamma(\widetilde{\phi_1}) \subseteq \gamma(\widetilde{\phi_2})$$

### 5.3 Abstraction

$$\alpha(\bar{\phi}) = \min_{\sqsubseteq} \{ \tilde{\phi} \mid \bar{\phi} \subseteq \gamma(\tilde{\phi}) \}$$

Proved:

- partial function
- sound
- optimal
- $\alpha(\gamma(\tilde{\phi})) = \tilde{\phi}$

### 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

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

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

$\hat{\phi}_m$  is evidence!

#### Consistent transitivity

While  $\Rightarrow$  is transitive,  $\widetilde{\Rightarrow}$  is generally not.

But maybe not even necessary with smarter hoare rules?

#### 5.4.3 Equality

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

$$\frac{\text{at least one of } \tilde{\phi}_1 \text{ or } \tilde{\phi}_2 \text{ contains ?} \quad \tilde{\phi}_1 \widetilde{\Rightarrow} \tilde{\phi}_2 \quad \tilde{\phi}_2 \widetilde{\Rightarrow} \tilde{\phi}_1}{\tilde{\phi}_1 \approx \tilde{\phi}_2} \text{GEQGRADUAL}$$



#### 5.4.4 Append

by definition:

$$\tilde{\phi} \tilde{*} \phi_p = \alpha(\gamma(\tilde{\phi}) \bar{*} \phi_p)$$

equivalent to:

$$\begin{array}{ll} \tilde{\phi} \tilde{*} \phi_p = \tilde{\phi} * \phi_p & \text{if } \forall \hat{\phi}_1, (\hat{\phi}_1 \implies \phi * \phi_p) \implies \exists \hat{\phi}_2, (\hat{\phi}_2 \implies \phi \wedge \hat{\phi}_1 \implies \hat{\phi}_2 * \phi_p) \\ & \text{if } \forall \hat{\phi}_1 \in \gamma(\tilde{\phi} * \phi_p), \exists \hat{\phi}_2 \in \gamma(\tilde{\phi}), \hat{\phi}_1 \implies \hat{\phi}_2 * \phi_p \\ \tilde{\phi} \tilde{*} \phi_p \text{ undefined} & \text{otherwise} \end{array}$$

### 5.5 Gradual Hoare: minimal static rule approach

Example:

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

Collapsing (hidden) gradual implications into a single one:

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

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:

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

#### 5.5.1 GHFieldAssign

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

#### 5.5.2 GHSec - sound but obviously not complete!

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

## 5.6 Gradual Hoare: minimal HSec approach (implications per rule)

$$\frac{\begin{array}{c} \vdash_{\text{sfrm}} \phi \quad \vdash C.f : T \\ \phi_1 \implies (x : C) * (y : T) * \phi * \text{acc}(x.f) \quad \phi_2 = (x : C) * \text{acc}(x.f) * (x.f = y) * \phi \end{array}}{\vdash \{\phi_1\}x.f := y\{\phi_2\}} \text{HFIELDASSIGN}$$

$$\frac{\begin{array}{c} \vdash_{\text{sfrm}} \phi \quad \vdash C.f : T \\ \widetilde{\phi}_1 \widetilde{\implies} (x : C) * (y : T) * \phi * \text{acc}(x.f) \quad \widetilde{\phi}_2 \approx (x : C) * \text{acc}(x.f) * (x.f = y) * \phi \end{array}}{\widetilde{\vdash} \{\widetilde{\phi}_1\}x.f := y\{\widetilde{\phi}_2\}} \text{GHFIELDASSIGN}$$

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

**Minimum runtime checks:** For  $\widetilde{\phi}_1 \widetilde{\implies} \widetilde{\phi}_2$  to hold at runtime, practically just  $\phi_2$  needs to hold. So that would be a valid assertion to check. Yet, we know statically that  $\phi_1$  holds, so we can remove everything from the runtime check that is implied by  $\phi_1$ . So in a sense, we only need to check  $\phi_2 \setminus \phi_1$  at runtime (the operator can be an approximation).

## 5.7 Gradual Hoare: deterministic approach

### 5.7.1 HFieldAssign

$$\frac{\begin{array}{c} \vdash C.f : T \\ \phi_1 \implies (x : C) * (y : T) * \text{acc}(x.f) \quad \phi_2 = (x : C) * \text{acc}(x.f) * (x.f = y) * \phi_1[\mathbf{w/o} \text{acc}(x.f)] \end{array}}{\vdash \{\phi_1\}x.f := y\{\phi_2\}} \text{HFIELDASSIGN}$$

Note:  $\phi[\mathbf{w/o} \text{acc}(x.f)]$  removes  $\text{acc}(x.f)$  and all uses of  $x.f$  from  $\phi$ . The result is self-framed given that  $\phi$  is.

**Attention:** This version is weaker than the other (pairwise equivalent) versions of HFieldAssign!

Explanation: Above operator may remove more information than necessary from  $\phi$ .

Example:

- Given:  $\phi_1 = \text{acc}(x.f) * (x.f = a) * (x.f = b)$
- Goal:  $\phi_2 \implies (a = b)$
- **not provable** with this deterministic version of HFieldAssign
- **provable** with all other versions

Probably it's possible to apply the operator without information loss after expanding formula using equalities (transitive hull).

### 5.7.2 GHFieldAssign

(= gradual lifting of GHFieldAssign as function)

$$\frac{\widetilde{\phi}_2 = \alpha(\{\phi_2 \mid \phi_1 \in \gamma(\widetilde{\phi}_1) \wedge \vdash \{\phi_1\}x.f := y\{\phi_2\}\})}{\widetilde{\vdash} \{\widetilde{\phi}_1\}x.f := y\{\widetilde{\phi}_2\}} \text{GHFIELDASSIGN}$$

Which should be equivalent to this:

$$\frac{\begin{array}{c} \vdash C.f : T \\ \phi_1 \implies (x : C) * (y : T) * \text{acc}(x.f) \\ \phi_2 = (x : C) * (y : T) * \text{acc}(x.f) * (x.f = y) * \phi_1[\mathbf{w/o} \text{acc}(x.f)] \end{array}}{\widetilde{\vdash} \{\phi_1\}x.f := y\{\phi_2\}} \text{GHFA1}$$

$$\frac{\begin{array}{c} \vdash C.f : T \\ ? * \phi_1 \widetilde{\Rightarrow}_{\phi_m} (x : C) * \mathbf{acc}(x.f) \\ \phi_2 = (x : C) * \mathbf{acc}(x.f) * (x.f = y) * \phi_m[\mathbf{w/o} \mathbf{acc}(x.f)] \end{array}}{\widetilde{\vdash} \{? * \phi_1\} x.f := y \{? * \phi_2\}} \text{GHFA2}$$

Which should be summarizable as this:

$$\frac{\begin{array}{c} \vdash C.f : T \\ \widetilde{\phi}_1 \widetilde{\Rightarrow}_{\widetilde{\phi}_m} (x : C) * (y : T) * \mathbf{acc}(x.f) \\ \widetilde{\phi}_2 = (x : C) * \mathbf{acc}(x.f) * (x.f = y) * \widetilde{\phi}_m[\mathbf{w/o} \mathbf{acc}(x.f)] \end{array}}{\widetilde{\vdash} \{\widetilde{\phi}_1\} x.f := y \{\widetilde{\phi}_2\}} \text{GHFA}$$

Which for well-formed programs is equivalent to:

$$\frac{\begin{array}{c} \vdash C.f : T \\ \phi_1 \Rightarrow (x : C) * (y : T) \quad \widetilde{\phi}_1 \widetilde{\Rightarrow} \mathbf{acc}(x.f) \\ \widetilde{\phi}_2 = (x : C) * (y : T) * \mathbf{acc}(x.f) * (x.f = y) * \widetilde{\phi}_1[\mathbf{w/o} \mathbf{acc}(x.f)] \end{array}}{\widetilde{\vdash} \{\widetilde{\phi}_1\} x.f := y \{\widetilde{\phi}_2\}} \text{GHFA}$$

Observations:

- $\widetilde{\phi}_m$  is the interior (first argument) of the implication, effectively the meet of first and second argument.
- for the gradual rules to work, the **w/o**-operator **must** be implemented with minimal information loss

## 5.8 Theorems

### 5.8.1 Soundness of $\alpha$

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

### 5.8.2 Optimality of $\alpha$

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

## 6 Theorems

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

#### 6.1.1 Phi valid

$$\vdash_{\mathbf{sfrm}} \phi$$

#### 6.1.2 Phi holds

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

#### 6.1.3 Types preserved

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

### 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

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

## 6.2 Soundness

### 6.2.1 Progress

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

### 6.2.2 Preservation

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