

1 Syntax

$program$	$::= \overline{cls} \ \bar{s}$
cls	$::= \text{class } C \ \{\overline{field} \ \overline{method}\}$
$field$	$::= T \ f;$
$method$	$::= T \ m(T \ x) \ contract \ \{\bar{s}\}$
$contract$	$::= \text{requires } \phi; \text{ ensures } \phi;$
T	$::= \text{int} \mid C$
s	$::= x.f := y; \mid x := e; \mid x := \text{new } C; \mid x := y.m(z);$ $\mid \text{return } x; \mid \text{assert } \phi; \mid \text{release } \phi; \mid T \ x;$
ϕ	$::= \text{true} \mid e = e \mid e \neq e \mid \text{acc}(e.f) \mid \phi * \phi$
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	$\in (o \multimap (C, (\overline{f \multimap v})))$
ρ	$\in (x \multimap v)$
Γ	$\in (x \multimap T)$
A_s	$::= \overline{(e, f)}$
A_d	$::= \overline{(o, f)}$
S	$::= (\rho, A_d, \bar{s}) \cdot S \mid nil$

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

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

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

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

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

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

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

$$\frac{\Gamma \vdash \{\hat{\phi}_p\} \bar{s}_1 \{\hat{\phi}_q\} \quad \Gamma \vdash \{\hat{\phi}_q\} \bar{s}_2 \{\hat{\phi}_r\}}{\Gamma \vdash \{\hat{\phi}_p\} \bar{s}_1; \bar{s}_2 \{\hat{\phi}_r\}} \text{HSEC}$$

3.5.2 Deterministic

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

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

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

Have $\hat{*}$ take care of necessary congruent rewriting of e in order to preserve self-framing!

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

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

$$\frac{\hat{\phi} \Longrightarrow \phi_a}{\Gamma \vdash \{\hat{\phi}\} \mathbf{assert} \phi_a \{\hat{\phi}\}} \text{HASSERTBAD (GRAD LIFTING NON-TRIVIAL!)}$$

$$\frac{\hat{\phi}[\mathbf{w/o} \lfloor \phi_a \rfloor] = \hat{\phi}' \quad \hat{\phi} \Longrightarrow \phi_a}{\Gamma \vdash \{\hat{\phi}\} \mathbf{assert} \phi_a \{\hat{\phi}' \hat{*} \phi_a\}} \text{HASSERT}$$

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

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

$$\frac{\Gamma \vdash \{\hat{\phi}_p\} s_1 \{\hat{\phi}_q\} \quad \Gamma \vdash \{\hat{\phi}_q\} \bar{s}_2 \{\hat{\phi}_r\}}{\Gamma \vdash \{\hat{\phi}_p\} s_1; \bar{s}_2 \{\hat{\phi}_r\}} \text{HSEC}$$

3.5.3 Gradual

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

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

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

Let $\tilde{*}$ behave like $\hat{*}$ if first operand is static - otherwise its regular concatenation.

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

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

$$\frac{\tilde{\phi}[\mathbf{w/o} \lfloor \phi_a \rfloor] = \tilde{\phi}' \quad \hat{\phi} \widetilde{\Longrightarrow} \phi_a}{\Gamma \vdash \{\tilde{\phi}\} \mathbf{assert} \phi_a \{\tilde{\phi}' \tilde{*} \phi_a\}} \text{GHASSERT}$$

$$\frac{\tilde{\phi}[\mathbf{w/o} \lfloor \phi_r \rfloor] = \tilde{\phi}' \quad \hat{\phi} \widetilde{\Longrightarrow} \phi_r}{\Gamma \vdash \{\tilde{\phi}\} \mathbf{release} \phi_r \{\tilde{\phi}'\}} \text{GHRELEASE}$$

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

$$\frac{\Gamma \vdash \{\tilde{\phi}_p\} s_1 \{\tilde{\phi}_q\} \quad \Gamma \vdash \{\tilde{\phi}_q\} \bar{s}_2 \{\tilde{\phi}_r\}}{\Gamma \vdash \{\tilde{\phi}_p\} s_1; \bar{s}_2 \{\tilde{\phi}_r\}} \text{GHSEC}$$

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

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

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

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

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

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

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

Note: 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”.

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

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

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

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

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

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

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

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

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

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

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

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

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 Self-framed and satisfiable formula

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

5.2 Concretization

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

$$\tilde{\phi}_1 \sqsubseteq \tilde{\phi}_2 : \iff \gamma(\tilde{\phi}_1) \subseteq \gamma(\tilde{\phi}_2)$$

5.3 Abstraction

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

Equivalent to:

$$\begin{aligned} \alpha(\{\phi\}) &= \phi \\ \alpha(\bar{\phi}) &= \dot{\alpha}(\bar{\phi}) := \sup_{\sqsubseteq} \{ ? * \phi \mid \phi \in \bar{\phi} \} \end{aligned}$$

Proved:

- partial function
- sound
- optimal
- $\alpha(\gamma(\tilde{\phi})) = \tilde{\phi}$
- does this make $\langle \gamma, \alpha \rangle$ a (partial) “galois insertion”?

5.4 Lifting functions

Gradual lifting $\tilde{f} : \tilde{\phi} \rightarrow \tilde{\phi}$ of a function $f : \phi \rightarrow \phi$:

$$\tilde{f}(\tilde{\phi}) := \alpha(\bar{f}(\gamma(\tilde{\phi})))$$

This formal definition has drawbacks:

- Calculations on infinite set (not implementable)
- Determine supremum of infinite set (not even clear if it exists)

Turns out above definition can be rewritten in an equivalent, computable way.

5.4.1 Dominator Theory

TODO: first tackle singleton case etc.

Theorem:

For every ϕ , there exists a finite set of “dominators” $\mathbf{dom}(\phi)$, such that

$$\gamma(? * \phi) = \bigcup_{\hat{\phi} \in \mathbf{dom}(f(\phi))} \gamma(? * \hat{\phi})$$

Consequence:

$$\begin{aligned} ? * \phi &= \alpha(\gamma(? * \phi)) \\ &= \dot{\alpha}(\gamma(? * \phi)) \\ &= \dot{\alpha}\left(\bigcup_{\hat{\phi} \in \mathbf{dom}(\phi)} \gamma(? * \hat{\phi})\right) \\ &= \dot{\alpha}\left(\bigcup_{\hat{\phi} \in \mathbf{dom}(\phi)} \{\hat{\phi}\}\right) \\ &= \dot{\alpha}(\mathbf{dom}(\phi)) \\ &= \sup_{\sqsubseteq} \{ ? * \phi' \mid \phi' \in \mathbf{dom}(\phi) \} \end{aligned}$$

Analogous, for monotonic f :

$$\begin{aligned} &\alpha(\bar{f}(\gamma(? * \phi))) \\ &= \dot{\alpha}(\bar{f}(\gamma(? * \phi))) \\ &= \dot{\alpha}(\bar{f}\left(\bigcup_{\hat{\phi} \in \mathbf{dom}(\phi)} \gamma(? * \hat{\phi})\right)) \\ &= \dot{\alpha}(\bar{f}\left(\bigcup_{\hat{\phi} \in \mathbf{dom}(\phi)} \{\hat{\phi}\}\right)) \\ &= \dot{\alpha}(\bar{f}(\mathbf{dom}(\phi))) \\ &= \sup_{\sqsubseteq} \{ ? * f(\phi') \mid \phi' \in \mathbf{dom}(\phi) \} \end{aligned}$$

Re-definition of gradual lifting:

$$\begin{aligned} \tilde{f}(\phi) &:= f(\phi) \\ \tilde{f}(? * \phi) &:= \alpha(\bar{f}(\gamma(? * \phi))) = \dot{\alpha}(\bar{f}(\mathbf{dom}(\phi))) \end{aligned}$$

In terms of implementation: At least no more infinite sets, need to calculating supremum remains.

Interesting observation:

$$\tilde{f}(? * \hat{\phi}) = \dot{\alpha}(\bar{f}(\mathbf{dom}(\hat{\phi}))) = \dot{\alpha}(\bar{f}(\{\hat{\phi}\})) = \dot{\alpha}(\{f(\hat{\phi})\}) = ? * f(\hat{\phi})$$

This observation raises the question whether it is possible to generalize the equality to work with arbitrary formulas, getting rid of $\dot{\alpha}$ and calculating a supremum entirely.

5.4.2 Generalization: Auto-liftable functions

Goal: Get a definition of \tilde{f} that is even easier to handle and implement. Therefore we want to investigate whether, or under which circumstances

$$\tilde{f}(\tilde{\phi}) = f(\tilde{\phi}) \quad (\text{i.e. } f \text{ applied to the static part of } \tilde{\phi})$$

holds.

We call functions f satisfying above equality “auto-liftable”.

Counterexamples:

- $f(\phi) = \text{acc}(x.f) * \phi$

$$\tilde{f}(? * x.f = 3) = ? * \text{false} \neq ? * \text{acc}(x.f) * (x.f = 3) = f(? * (x.f = 3))$$

Cause: $\gamma(? * x.f = 3)$ only contains self-framed formulas, so access to $x.f$ is always included. Adding it another time results in duplicate access and therefore unsatisfiable formulas.

- $f(\phi) = \text{remove all terms containing } x$

$$\tilde{f}(? * a = 3) = ? \neq ? * (a = 3) = f(? * (a = 3))$$

Cause: $(a = x) * (x = 3) \in \gamma(? * a = 3)$ and $f((a = x) * (x = 3)) = \text{true}$. Abstracting from a (non-singleton) set that contains **true** yields $?$.

What is necessary to generalize this as $\alpha(\bar{f}(\gamma(? * \phi))) = ? * f(\phi)$?

$$\begin{aligned} & \forall \phi' \in \gamma(? * f(\phi)), \exists \phi'' \in \gamma(? * \phi), \phi' \in \gamma(? * f(\phi'')) \\ & \implies \\ & \forall \phi' \in \gamma(? * f(\phi)), \exists \phi'' \in \gamma(? * \phi), ? * \phi' \sqsubseteq ? * f(\phi'') \\ & \implies \\ & \forall \phi' \in \text{dom}(f(\phi)), \exists \phi'' \in \text{dom}(\phi), ? * \phi' \sqsubseteq ? * f(\phi'') \\ & \implies \\ & \forall \phi' \in \text{dom}(f(\phi)), ? * \phi' \sqsubseteq \sup_{\sqsubseteq} \{ ? * f(\phi') \mid \phi' \in \text{dom}(\phi) \} \\ & \iff \\ & \sup_{\sqsubseteq} \{ ? * \phi' \mid \phi' \in \text{dom}(f(\phi)) \} \sqsubseteq \sup_{\sqsubseteq} \{ ? * f(\phi') \mid \phi' \in \text{dom}(\phi) \} \\ & \iff \\ & ? * f(\phi) \sqsubseteq \alpha(\bar{f}(\gamma(? * \phi))) \\ & \\ & ? * f(\phi) \sqsubseteq ? * f(\phi) \\ & \implies \\ & \forall \phi' \in \text{dom}(\phi), ? * f(\phi') \sqsubseteq ? * f(\phi) \\ & \iff \\ & \sup_{\sqsubseteq} \{ ? * f(\phi') \mid \phi' \in \text{dom}(\phi) \} \sqsubseteq ? * f(\phi) \\ & \iff \\ & \alpha(\bar{f}(\gamma(? * \phi))) \sqsubseteq ? * f(\phi) \end{aligned}$$

For a function f to be auto-liftable, the following properties are sufficient:

- Monotonicity
- $\forall \phi' \in \gamma(? * f(\phi)), \exists \phi'' \in \gamma(? * \phi), \phi' \in \gamma(? * f(\phi''))$

5.4.3 Lifiable composition

Given liftable functions f and g , is $g \circ f$ liftable? Monotonicity is obviously preserved. Other condition:

$$\begin{aligned}
& ? * g(f(\phi)) \sqsubseteq \alpha(\bar{g}(\gamma(? * f(\phi)))) \wedge ? * f(\phi) \sqsubseteq \alpha(\bar{f}(\gamma(? * \phi))) \\
& \implies \\
& ? * g(f(\phi)) \sqsubseteq \alpha(\bar{g}(\gamma(? * f(\phi)))) \wedge \alpha(\gamma(? * f(\phi))) \sqsubseteq \alpha(\bar{f}(\gamma(? * \phi))) \\
& \implies \\
& ? * g(f(\phi)) \sqsubseteq \alpha(\bar{g}(\gamma(? * f(\phi)))) \wedge \alpha(\bar{g}(\gamma(? * f(\phi)))) \sqsubseteq \alpha(\bar{g}(\bar{f}(\gamma(? * \phi)))) \\
& \implies \\
& ? * g(f(\phi)) \sqsubseteq \alpha(\bar{g}(\bar{f}(\gamma(? * \phi)))) \\
& \implies \\
& ? * (g \circ f)(\phi) \sqsubseteq \alpha(\overline{(g \circ f)}(\gamma(? * \phi)))
\end{aligned}$$

5.5 Gradual Lifting

5.5.1 Self framing

$$\frac{A \vdash_{\text{sfrm}} \phi}{A \vdash_{\text{sfrm}} \phi} \text{GSFRMNONGRAD}$$

$$\frac{}{A \vdash_{\text{sfrm}} ? * \phi} \text{GSFRMGRAD}$$

5.5.2 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?

5.5.3 Equality

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

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

5.5.4 Append

by definition:

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

equivalent to:

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

5.6 Gradual Hoare: minimal static rule approach

Example:

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

Collapsing (hidden) gradual implications into a single one:

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

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:

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

5.6.1 GHFieldAssign

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

5.6.2 GHSec - sound but obviously not complete!

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

5.7 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 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 $\phi_2 \setminus \phi_1$ at runtime (the operator can be an approximation).

5.8 Gradual Hoare: deterministic approach

5.8.1 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 ϕ . The result is self-framed given that ϕ is.

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

Explanation: Above operator may remove more information than necessary from ϕ .

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).

5.8.2 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

5.9 Theorems

5.9.1 Soundness of α

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

5.9.2 Optimality of α

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

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

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

6.2 Soundness

6.2.1 Progress

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

6.2.2 Preservation

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