

1 Syntax

<i>program</i>	$::= \overline{cls} \ \overline{s}$
<i>cls</i>	$::= \text{class } C \ \{\overline{field} \ \overline{method}\}$
<i>field</i>	$::= T \ f;$
<i>method</i>	$::= T \ m(T \ x) \ \text{contract } \{\overline{s}\}$
<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	$::= \text{true} \mid e = e \mid e \neq e \mid \text{acc}(x.f) \mid x : T \mid \phi * \phi$
<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>	$::= \overline{(x, f)}$
<i>A_d</i>	$::= \overline{(o, f)}$
<i>S</i>	$::= (\rho, A_d, \overline{s}) \cdot S \mid \text{nil}$

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)

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

2.0.2 Well-formed class (*cls* OK)

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

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

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

3 Static semantics

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$\lfloor \mathbf{true} \rfloor$	$= \emptyset$
$\lfloor e_1 = e_2 \rfloor$	$= \emptyset$
$\lfloor e_1 \neq e_2 \rfloor$	$= \emptyset$
$\lfloor \mathbf{acc}(x.f) \rfloor$	$= \{(x, f)\}$
$\lfloor \phi_1 * \phi_2 \rfloor$	$= \lfloor \phi_1 \rfloor \cup \lfloor \phi_2 \rfloor$

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

$$\frac{}{\phi \vdash v_T : T} \text{STVALUE}$$

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

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

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

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

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

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

$$\frac{\phi_1 \vdash x : T \quad \phi_1 \vdash e : T \quad \phi_1 = \phi_2[e/x] \quad \emptyset \vdash_{\text{sfrm}} \phi_1 \quad \lfloor \phi_1 \rfloor \vdash_{\text{sfrm}} e}{\vdash \{\phi_1\} x := e \{\phi_2\}} \text{HVARASSIGN}$$

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

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

$$\frac{\phi_1 \implies \phi_2}{\vdash \{\phi_1\} \mathbf{assert} \phi_2 \{\phi_1\}} \text{HASSERT}$$

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

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

4 Dynamic semantics

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

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

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

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

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

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

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

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

$$\frac{\rho(x) = o \quad (o, f) \in A}{H, \rho, A \models \mathbf{acc}(x.f)} \text{EAACC}$$

$$\frac{\rho(x) = v_T}{H, \rho, A \models x : T} \text{EATYPE}$$

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

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 ($\phi_1 \implies \phi_2$)

$$\phi_1 \implies \phi_2 \iff \llbracket \phi_1 \rrbracket \subseteq \llbracket \phi_2 \rrbracket$$

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

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

4.4 Type ($H, \rho \vdash e : T$)

$$\frac{H, \rho \vdash e \Downarrow v_T}{H, \rho \vdash e : T} \text{DTEVAL}$$

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

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

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

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

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

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

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

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

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

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

5 Gradualization

5.1 Syntax

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

Note: allowing $?$ at different position is hardly useful

- dynamically: order makes no difference
- statically: Imagine substituting $?$ with $\text{acc}(x.f) * [\dots]$. The only time this makes sense to **not** put to the very right is when there are expressions containing $x.f$ (in the non-gradual part). In other words, there are expressions that are framed only when $?$ is substituted in a specific way. But then why not require the necessary framing in the non-gradual part as well, i.e. why not simply **write out** $\text{acc}(x.f)$ instead of relying on a substitution? Corollary: The non-gradual part of $\tilde{\phi}$ should be self-framed.

5.1.1 Discussion

- We want $\phi * ? \implies \phi$ for all ϕ
- Meaning of $\vdash \{\phi_1 * ?\} \bar{s} \{\phi_2\}$ compared to $\vdash \{\phi_1\} \bar{s} \{\phi_2\}?$
 - Caller: nothing
 - Callee: verification succeeds as long as there exists a (satisfiable) instantiation that makes proof about method body work.
- Meaning of $\vdash \{\phi_1\} \bar{s} \{\phi_2 * ?\}$ compared to $\vdash \{\phi_1\} \bar{s} \{\phi_2\}?$
 - Caller: verification succeeds as long as there exists a (satisfiable) instantiation that makes upcoming proofs work.
 - Callee: nothing

5.2 Concretization A

$$\begin{aligned}
\gamma(\phi) &= \{ \phi' \mid \phi' \iff \phi \} \\
\gamma(\phi * ?) &= \{ \phi' \mid \exists \phi_x : \phi * \phi_x \iff \phi' \} \\
&= \{ \phi' \mid \phi' \implies \phi \}
\end{aligned}$$

5.3 Abstraction (to show: (ϕ, \implies) is lattice)

$$\alpha(\bar{\phi}) = (\bigcap \bar{\phi}) * ?$$

5.4 Concretization B (as in better)

$$\begin{aligned}
\gamma(\phi) &= \{ \phi \} \\
\gamma(\phi * ?) &= \{ \phi * \phi_x \mid \exists \phi_x : \llbracket \phi * \phi_x \rrbracket \neq \emptyset \}
\end{aligned}$$

5.5 Abstraction (to show: (ϕ, \implies) is lattice)

$$\begin{aligned}
\alpha(\{ \phi \}) &= \phi \\
\alpha(\bar{\phi}) &= (\bigcap \bar{\phi}) * ?
\end{aligned}$$

Note: $\gamma(\phi * ?)$ contains $\phi * \mathbf{true}$ which is obviously implied by all the other members of the set, so $\bigcap \gamma(\phi * ?) = \phi * \mathbf{true} * ?$.

It feels like the operations on ϕ -sets we will encounter will preserve this property of $\bigcap \bar{\phi}$ actually being a well-known member of $\bar{\phi}$.

5.6 Theorems

5.6.1 Soundness of α

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

5.6.2 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 $\text{invariant}(H, \rho, A_d, \phi)$

6.1.1 Heap consistent

$$\begin{aligned} \forall x, o, C : \rho(x) = o_C &\implies \\ \exists f_C, m : \text{fields}(C) = f_C & \\ \wedge H(o_C) = (C, m) & \\ \wedge (\forall (T, f) \in f_C : H, \rho \vdash m(f) : T) & \end{aligned}$$

6.1.2 Phi self-framed

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

6.1.3 Phi holds

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

6.1.4 Types preserved

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

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

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