1 Syntax

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

2 Static semantics

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

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

$$\frac{}{A \vdash_{\texttt{sfrm}} v}\ \textsc{WFValue}$$

$$\frac{(x, f) \in A}{A \vdash_{\texttt{sfrm}} x.f}\ \textsc{WFField}$$

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

$$\frac{}{A \vdash_{\texttt{sfrm}} \texttt{true}}\ \textsc{WFTrue}$$

$$\frac{A \vdash_{\texttt{sfrm}} e_1 \qquad A \vdash_{\texttt{sfrm}} e_2}{A \vdash_{\texttt{sfrm}} (e_1 = e_2)} \textsc{WFEqual}$$

$$\frac{A \vdash_{\texttt{sfrm}} e_1 \qquad A \vdash_{\texttt{sfrm}} e_2}{A \vdash_{\texttt{sfrm}} (e_1 \neq e_2)} \textsc{WFNEqual}$$

$$\frac{}{A \vdash_{\texttt{sfrm}} \mathsf{acc}(x.f)} \textsc{WFAcc}$$

$$\frac{}{A \vdash_{\texttt{sfrm}} x : T} \textsc{WFType}$$

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

2.2.1 Implication ($\phi_1 \overset{.}{\implies} \phi_2$)

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

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

$$\begin{aligned}
\lfloor \texttt{true} \rfloor &= \emptyset \\
\lfloor e_1 = e_2 \rfloor &= \emptyset \\
\lfloor e_1 \neq e_2 \rfloor &= \emptyset \\
\lfloor \mathsf{acc}(x.f) \rfloor &= \{(x, f)\} \\
\lfloor \phi_1 * \phi_2 \rfloor &= \lfloor \phi_1 \rfloor \cup \lfloor \phi_2 \rfloor
\end{aligned}$$

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

$$\frac{}{\phi \vdash v_T : T} \textsc{STValue}$$

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

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

2.5 Hoare ($\vdash \{\phi\}\overline{s}\{\phi\}$)

$$\frac{\vdash \{\phi_p\}s_1\{\phi_{q1}\} \qquad \phi_{q1} \implies \phi_{q2} \qquad \vdash \{\phi_{q2}\}s_2\{\phi_r\}}{\vdash \{\phi_p\}s_1; s_2\{\phi_r\}}\ \textsc{HSec}$$

$$\frac{\phi \vdash x : C \qquad \texttt{fields}(C) = \overline{f}}{\vdash \{\phi\}x := \texttt{new}\ C\{\overline{\texttt{acc}(x, f_i)*}(x \neq \texttt{null}) * \phi\}}\ \textsc{HNewObj}$$

$$\frac{\phi \vdash x : C \qquad \vdash C.f : T \qquad \phi \vdash y : T \qquad \phi \implies \texttt{acc}(x.f) \qquad \phi \implies (x \neq \texttt{null})}{\vdash \{\phi\}x.f := y\{\phi * (x.f = y)\}}\ \textsc{HFieldAssign}$$

$$\frac{\phi_1 \vdash x : T \qquad \phi_1 \vdash e : T \qquad \phi_1 = \phi_2[e/x] \qquad \emptyset \vdash_{\texttt{sfrm}} \phi_1 \qquad \lfloor \phi_1 \rfloor \vdash_{\texttt{sfrm}} e}{\vdash \{\phi_1\}x := e\{\phi_2\}}\ \textsc{HVarAssign}$$

$$\frac{\phi \vdash x : T \qquad \phi \vdash \texttt{result} : T}{\vdash \{\phi\}\texttt{return}\ x\{\phi * (\texttt{result} = x)\}}\ \textsc{HReturn}$$

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

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

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

$$\frac{x \notin \texttt{dom}(\phi_1) \qquad \phi_2 = \phi_1 * x : T * (x = \texttt{defaultValue}(T))}{\vdash \{\phi_1\}T\ x\{\phi_2\}}\ \textsc{HDeclare}$$

3 Dynamic semantics

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

$$\frac{}{H, \rho \vdash x \Downarrow \rho(x)}\ \textsc{EEVar}$$

$$\frac{}{H, \rho \vdash v \Downarrow v}\ \textsc{EEValue}$$

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

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

$$\frac{}{H, \rho, A \vDash \texttt{true}}\ \textsc{EATrue}$$

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

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

$$\frac{\rho(x) = o \qquad (o, f) \in A}{H, \rho, A \vDash \texttt{acc}(x.f)}\ \textsc{EAAcc}$$

$$\frac{\rho(x) = v_T}{H, \rho, A \vDash x : T}\ \textsc{EAType}$$

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

3.2.1 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”.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4 Gradualization

4.1 Syntax

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

Note: allowing ? at different position is hardly useful (no difference for dyn. semantics, useless difference in static semantics).

4.2 Concretization A

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

4.3 Abstraction (to show: set of ϕs is poset - can we even say that on infinite sets?)

$$\alpha(\overline{\phi}) = (\sqcap \overline{\phi}) * ?$$

4.4 Concretization B

$$\begin{aligned}
\gamma(\phi) \quad &= \{\ \phi\ \} \\
\gamma(\phi * ?) \quad &= \{\ \phi * \phi_x \mid \phi_x\ \}
\end{aligned}$$

4.5 Abstraction

$$\alpha(\{\ \phi\ \}) = \phi$$
$$\alpha(\{\ \overline{\phi}\ \}) = \mathtt{lcp}(\overline{\phi})$$

4.6 Theorems

4.6.1 Soundness

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

4.6.2 Optimality

$$\forall \overline{\phi}, \widetilde{\phi} : \overline{\phi} \subseteq \gamma(\widetilde{\phi}) \implies \gamma(\widetilde{\phi}) \subseteq \gamma(\alpha(\overline{\phi}))$$

4.7 Concretization W (as in wrong)

$$\begin{aligned}
\gamma(\phi) &= \{\ (H, \rho, A) \mid H, \rho, A \vDash \phi\ \} \\
\gamma(\phi * ?) &= \{\ (H, \rho, A) \mid H, \rho, A \vDash \phi\ \}
\end{aligned}$$

5 Theorems

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

5.1.1 Heap consistent

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

5.1.2 Phi holds

$$H, \rho, A_d \vDash \phi$$

5.1.3 Types preserved

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

5.2 Soundness

5.2.1 Progress

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

5.2.2 Preservation

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