1 Syntax

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

2 Assumptions

All the rules in the following sections are implicitly parameterized over a *programp* that is well-formed.

2.0.1 Well-formed program ($\mathit{program}\ \texttt{OK}$)

$$\frac{\overline{\mathit{cls}_i\ \texttt{OK}}}{(\overline{\mathit{cls}_i}\ \overline{s})\ \texttt{OK}}\ \textsc{OKProgram}$$

2.0.2 Well-formed class ($\mathit{cls}\ \texttt{OK}$)

$$\frac{\text{unique } \mathit{field}\text{-names} \qquad \text{unique } \mathit{method}\text{-names} \qquad \overline{\mathit{method}_i\ \texttt{OK in}\ C}}{(\texttt{class}\ C\ \{\overline{\mathit{field}_i}\ \overline{\mathit{method}_i}\})\ \texttt{OK}}\ \textsc{OKClass}$$

2.0.3 Well-formed method ($\mathit{method}\ \texttt{OK in}\ C$)

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

3 Static semantics

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

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

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

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

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

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

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

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

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

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

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

3.2.1 Implication ($\phi_1 \stackrel{.}{\Longrightarrow} \phi_2$)

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

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

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

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

$$\frac{}{\phi \vdash n : \texttt{int}} \textsc{STValNum}$$

$$\frac{}{\phi \vdash \texttt{null} : T} \textsc{STValNull}$$

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

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

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

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

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

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

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

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

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

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

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

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

Note: issue with HApp and z' in the post-condition: the substitution reflects **any changes** made to z onto z' which is wrong in general (except we make z' a by-ref parameter in the small-step semantics)

4 Dynamic semantics

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

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

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

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

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

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

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

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

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

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

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

We give a denotational semantics of formulas as $[\![\phi]\!] = \{\ (H, \rho, A) \mid H, \rho, A \vDash \phi\ \}$
Note: ϕ satisfiable $\iff [\![\phi]\!] \neq \emptyset$

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

$$\phi_1 \implies \phi_2 \qquad \iff \qquad [\![\phi_1]\!] \subseteq [\![\phi_2]\!]$$

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

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

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

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

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

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

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

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

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

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

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

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

5 Gradualization

5.1 Syntax

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

Note: allowing ? at different position is hardly useful

- dynamically: order makes no difference
- statically: Imagine substituting ? with $acc(x.f) * [...]$. 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** $acc(x.f)$ instead of relying on a substitution? Corollary: The non-gradual part of $\widetilde{\phi}$ should be self-framed.

5.1.1 Discussion

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

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

5.4 Concretization B (as in better)

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

5.5 Abstraction

$$\begin{aligned}
\alpha(\{\ \phi\ \}) &= \phi \\
\alpha(\overline{\phi}) &= (\bigsqcap \overline{\phi}) * ?
\end{aligned}$$

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

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

5.6 $(\phi, \implies)$ is semilattice

According to the definition of $\implies$ via $\subseteq$, we define

$$\phi_a \sqcap \phi_b = \phi_c \quad :\Longleftrightarrow \quad [\![\phi_a]\!] \cap [\![\phi_b]\!] = [\![\phi_c]\!]$$

The question is, whether such ϕ_c always exists.

5.7 Theorems

5.7.1 Soundness of α

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

5.7.2 Optimality of α

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

6 Theorems

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

6.1.1 Phi valid

$$\vdash_{\texttt{sfrm}} \phi$$

6.1.2 Phi holds

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

6.1.3 Types preserved

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

6.1.4 Heap consistent

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

6.1.5 Heap not total

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

6.2 Soundness

6.2.1 Progress

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

6.2.2 Preservation

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