1 Syntax

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

2 Assumptions

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

2.0.1 Well-formed program (*program* OK)

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

2.0.2 Well-formed class (*cls* OK)

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

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

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

3 Static semantics

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

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

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

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

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

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

$$\frac{A \vdash_{\mathtt{sfrm}} e_1 \qquad A \vdash_{\mathtt{sfrm}} e_2}{A \vdash_{\mathtt{sfrm}} (e_1 = e_2)} \; \text{WFEQUAL}$$

$$\frac{A \vdash_{\mathtt{sfrm}} e_1 \qquad A \vdash_{\mathtt{sfrm}} e_2}{A \vdash_{\mathtt{sfrm}} (e_1 \neq e_2)} \; \text{WFNEQUAL}$$

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

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

$$\frac{A_s \vdash_{\mathtt{sfrm}} \phi_1 \qquad A_s \cup \lfloor \phi_1 \rfloor \vdash_{\mathtt{sfrm}} \phi_2}{A_s \vdash_{\mathtt{sfrm}} \phi_1 * \phi_2} \; \text{WFSEPOP}$$

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

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 ($\phi \vdash e : T$)

$$
\frac{}{\phi \vdash n : \texttt{int}} \ \text{STVALNUM}
$$

$$
\frac{}{\phi \vdash \texttt{null} : T} \ \text{STVALNULL}
$$

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

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

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

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

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

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

$$
\frac{\phi \implies \phi' \qquad \emptyset \vdash_{\texttt{sfrm}} \phi' \qquad x \notin FV(\phi') \qquad \phi \vdash x : T \qquad \phi \vdash e : T \qquad \lfloor \phi' \rfloor \vdash_{\texttt{sfrm}} e}{\vdash \{\phi\}x := e\{\phi' * (x = e) * \emptyset\}} \ \text{HVARASSIGN}
$$

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

3

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\rho(x) = o \qquad (o, f) \in A}{H, \rho, A \vDash \mathtt{acc}(x.f)} \text{ EAACC}$$

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

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

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

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

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

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 ($(H, S) \to (H, S)$)

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\forall e, T : \phi \vdash e : T$$
$$\implies H, \rho \vdash e : T$$

6.1.4 Heap consistent

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

6.1.5 Heap not total

$$\exists o_{min} :$$
$$\forall o \geq o_{min} : o \notin \texttt{dom}(H)$$
$$\wedge \; \forall f, (o, f) \notin A$$

6.2 Soundness

6.2.1 Progress

$$\forall \ldots : \quad \vdash \{\phi_1\} s' \{\phi_2\}$$
$$\implies invariant(H_1, \rho_1, A_1, \phi_1)$$
$$\implies \exists H_2, \rho_2, A_2 : (H_1, (\rho_1, A_1, s'; \overline{s}) \cdot S) \to^* (H_2, (\rho_2, A_2, \overline{s}) \cdot S)$$

6.2.2 Preservation

$$\forall \ldots : \quad \vdash \{\phi_1\} s' \{\phi_2\}$$
$$\implies invariant(H_1, \rho_1, A_1, \phi_1)$$
$$\implies (H_1, (\rho_1, A_1, s'; \overline{s}) \cdot S) \to^* (H_2, (\rho_2, A_2, \overline{s}) \cdot S)$$
$$\implies invariant(H_2, \rho_2, A_2, \phi_2)$$