Coq formalization

Billy Bai

February 2024

1 Whole setting

1.1 Syntax

$$
\begin{array}{rcl}
P & ::= & S \\
ct & ::= & \overline{ClassDecl} \\
\text{ClassDecl} & ::= & \text{class } C \ \{ \ \overline{F} \ K \ \overline{M} \ \} \\
F & ::= & f : \ T; \\
K & ::= & C \ (\overline{p : T}) \ : \text{ret} : \rho \ C : \ \{ \ \overline{\text{this}.f := p;} \ \text{ret} := \mathit{this} \ \} \\
M & ::= & \text{def } m(\text{this} : \rho \ C, \ \overline{z : T}) \ : \text{ret} : T : \ \{S\} \\
e & ::= & x \mid c \mid x.f \\
S & ::= & \text{skip} \mid x := y \mid x := y.f \mid x.f := y \mid x := y.m(\overline{z}) \\
 & \mid & \text{var } x : T := e \ \mathsf{in} \ S \mid \text{var } x : C := \text{new } \rho \ C(\overline{y}) \ \mathsf{in} \ S \mid \mathsf{if} \ e \ \text{then} \ S \ \mathsf{else} \ S \\
 & \mid & \mathsf{while} \ e \ \mathsf{do} \ S \ in \ c \mid S; \ S \\
T & ::= & \text{Bool} \mid \rho \ \text{C} \\
\rho & ::= & \text{Shared} \mid \text{Unique} \mid \bot \\
c & ::= & True \mid False
\end{array}
$$

1.2 Runtime Model

$$
\begin{array}{rcl}
v & ::= & c \mid \&l \\
c & ::= & True \mid False \\
l & \in & \mathbb{N} \\
h & ::= & \overline{l \mapsto (\rho \ C, \overline{fv})} \\
fv & ::= & v \\
\sigma & ::= & \overline{x := (T, v)}
\end{array}
$$

2 Well-Formness

$\boxed{WF(ct)}$

$$\frac{}{\mathrm{WF}(\emptyset)}\ (\textsc{wf-ct-nil}) \qquad \frac{\mathrm{WF}(ct) \quad \mathit{ClassDecl} = \text{class } C \ \{\ \overline{F}\ K\ \overline{M}\ \} \quad WF_{field}(\overline{F}, ct) \quad WF_{method}(\overline{M}, ct)}{\mathrm{WF}(\mathit{ClassDecl}_k :: ct)}\ (\textsc{wf-ct-nil})$$

$\boxed{WF_{field}(\overline{F}, ct)}$

$$\frac{}{\mathrm{WF}_{field}(\emptyset, ct)}\ (\textsc{wf-f-nil}) \qquad \frac{\mathrm{WF}_{field}(\overline{F}, ct) \quad WF_{ct}(T_f) \quad T_f = \rho\ C \implies \rho = \mathit{Shared}}{\mathrm{WF}_{field}(f \mapsto T_f :: \overline{F}, ct)}\ (\textsc{wf-f-cons}) \qquad \frac{WF_{ct}(T_f) \quad T_f = \rho\ C \implies \rho = \mathit{Shared}}{\mathrm{WF}_{field}(f \mapsto T_f, ct)}\ (\textsc{wf-f-alter})$$

$\boxed{WF_{method}(\overline{M}, ct)}$

$$\frac{}{\mathrm{WF}_{method}(\emptyset, ct)}\ (\textsc{wf-m-nil}) \qquad \frac{WF_{method}(\overline{M}) \quad WF_{ct}(\rho\ C) \quad WF_{ct}(\overline{T}) \quad WF_{ct}(T_r)}{\mathrm{WF}_{method}((m \mapsto m(\mathit{this} : \rho\ C, \overline{z : T}) : \mathit{ret} : T_r : \{S\}) :: \overline{M}, ct)}\ (\textsc{wf-m-cons})$$

$$\frac{WF_{ct}(\rho\ C) \quad WF_{ct}(\overline{T}) \quad WF_{ct}(T_r)}{\mathrm{WF}_{method}((m \mapsto m(\mathit{this} : \rho\ C, \overline{z : T}) : \mathit{ret} : T_r : \{S\}), ct)}\ (\textsc{wf-m-alter})$$

$\boxed{WF_{ct}(T)}$

$$\frac{}{\mathrm{WF}_{ct}(\mathit{Bool})}\ (\textsc{wf-t-Bool}) \qquad \frac{\mathrm{ClassDecl} \in ct \quad \rho \neq \bot}{\mathrm{WF}_{ct}(\rho\ C)}\ (\textsc{wf-t-class})$$

Figure 1: Definition of well-formness.

3 Typing Rule

$\boxed{\Gamma \vdash e : \ T}$

$$\frac{}{\Gamma \vdash c : \text{Bool}}\ (\text{T-c}) \qquad \frac{\Gamma(x) = T \quad \text{WF}_{ct}(T)}{\Gamma \vdash x : T}\ (\text{T-var})$$

$$\frac{\Gamma(x) = \text{C}_i \quad \text{lookup}_{ct}(C_i, f) = T \quad \text{WF}_{ct}(T)}{\Gamma \vdash x.f : T}\ (\text{T-facc})$$

Figure 2: Type rules for expressions.

$\boxed{\Gamma \vdash s \dashv \Gamma'}$

$$\frac{}{\Gamma \vdash \mathit{skip} :\dashv \Gamma}\ (\text{T-skip}) \qquad \frac{\Gamma \vdash x : \mathit{Bool} \quad \Gamma \vdash y : \mathit{Bool}}{\Gamma \vdash x := y :\dashv \Gamma}\ (\text{T-assign-C})\ (\text{omit later})$$

$$\frac{\Gamma \vdash x : \rho\ C \quad \Gamma \vdash y : \mathit{Shared}\ C}{\Gamma \vdash x := y :\dashv \Gamma[x \mapsto \mathit{Shared}\ C]}\ (\text{T-assign-S}) \qquad \frac{\Gamma \vdash x : \rho\ C \quad \Gamma \vdash y : \mathit{Unique}\ C}{\Gamma \vdash x := y :\dashv \Gamma[y \mapsto \bot C][x \mapsto \mathit{Unique}\ C]}\ (\text{T-assign-U})$$

$$\frac{\Gamma \vdash x : \rho\ C \quad \Gamma \vdash y.f : \mathit{Shared}\ C}{\Gamma \vdash x := y.f \dashv \Gamma[x \mapsto \mathit{Shared}\ C]}\ (\text{T-facc-S})$$

$$\frac{\begin{array}{c}\Gamma \vdash x.f : \mathit{Shared}\ C \\ \Gamma \vdash y : \mathit{Shared}\ C \quad \Gamma \vdash x : \mathit{Shared}\ C'\end{array}}{\Gamma \vdash x.f := y \dashv \Gamma}\ (\text{T-fupd-S}) \qquad \frac{\begin{array}{c}\Gamma \vdash x.f : \mathit{Shared}\ C \\ \Gamma \vdash y : \mathit{Unique}\ C \quad \Gamma \vdash x : \mathit{Shared}\ C'\end{array}}{\Gamma \vdash x.f := y \dashv \Gamma[y \mapsto \bot\ C]}\ (\text{T-fupd-U})$$

$$\frac{\begin{array}{c}\text{ClassDecl} = \text{Class}\ C\{\ \overline{F}\ K\ \overline{M}\} \\ M(m) = \text{def}\ m(\text{this} : C,\ \overline{p : T}) : \text{ret} : T_r : \{S\} \\ \Gamma \vdash x : T_r \quad \Gamma \vdash y : C \quad \Gamma \vdash \overline{z} : \overline{T} \\ \Gamma \vdash S[y/\mathit{this}, \overline{z}/\overline{p}] \dashv \Gamma' \quad \Gamma' \vdash \mathit{ret}[y/\mathit{this}, \overline{z}/\overline{p}] : T_r\end{array}}{\Gamma \vdash x := y.m(\overline{z}) \dashv \Gamma''}\ (\text{T-MCall})$$

Figure 3: Type rules for statements(1).

$$\boxed{\Gamma \vdash s \dashv \Gamma'}$$

$$\text{(T-LETTERM-S)}\quad \frac{\mathit{FreeVar}^s(S,\Gamma) = x \quad \Gamma \vdash e : \mathit{Shared}\ C \qquad \Gamma, x \mapsto \mathit{Shared}\ C \vdash S \dashv \Gamma', x \mapsto \rho\ C}{\Gamma \vdash \mathit{var}\ x : \mathit{Shared}\ C := e\ \mathit{in}\ S \dashv \Gamma'}$$

$$\text{(T-LETTERM-U)}\quad \frac{\mathit{FreeVar}^s(S,\Gamma) = x \quad \Gamma \vdash y : \mathit{Unique}\ C \qquad \Gamma[y \mapsto \bot\ C], x \mapsto \mathit{Unique}\ C \vdash S \dashv \Gamma', x \mapsto \rho\ C}{\Gamma \vdash \mathit{var}\ x : \mathit{Unique}\ C := y\ \mathit{in}\ S \dashv \Gamma'}$$

$$\text{(T-LETNEW)}\quad \frac{\begin{array}{c}\text{ClassDecl} = \text{Class}\ C\{\ \overline{F}\ K\ \overline{M}\} \\ K = C\ (\overline{p : T_f})\ : \text{ret} : \rho\ C :\ \{\overline{\text{this}.f := p};\ \text{ret} := \mathit{this}\} \\ \mathit{FreeVar}^s(S,\Gamma) = x \quad \Gamma \vdash \overline{y} : \overline{T_f} \quad \Gamma, x \mapsto \rho\ C \vdash S \dashv \Gamma', x \mapsto \rho'\ C\end{array}}{\Gamma \vdash \mathit{var}\ x : \rho\ C := \mathit{new}\ C_i(\overline{y})\ \mathit{in}\ S \dashv \Gamma'}$$

$$\text{(T-IF)}\quad \frac{\Gamma \vdash e : \mathit{Bool} \quad \Gamma \vdash S_1 \dashv \Gamma_1 \quad \Gamma \vdash S_2 \dashv \Gamma_2}{\Gamma \vdash \mathit{if}\ e\ \mathit{then}\ S_1\ \mathit{else}\ S_2 \dashv \Gamma_1 \bowtie \Gamma_2}$$

$$\text{(T-LOOP)}\quad \frac{\Gamma \vdash e : \mathit{Bool} \quad \Gamma \vdash\ S^c \dashv \Gamma'}{\Gamma \vdash \mathit{while}\ e\ \mathit{do}\ S\ \mathit{in}\ c \dashv \Gamma'}$$

$$\text{(T-SEQ)}\quad \frac{\Gamma \vdash S_1 \dashv \Gamma' \quad \Gamma' \vdash S_2 \dashv \Gamma'}{\Gamma \vdash S_1; S_2 \dashv \Gamma''}$$

Figure 4: Type rules for statements(2).

4 Semantics

$$\boxed{(h;\sigma;e) \rightsquigarrow (T,v)}$$

$$\text{(S-C)}\quad \frac{}{(h;\sigma;c) \rightsquigarrow (\mathit{Bool}, c)}$$

$$\text{(S-VAR)}\quad \frac{\sigma(x) = (T,v)}{(h;\sigma;x) \rightsquigarrow (T,v)}$$

$$\text{(S-VFACC)}\quad \frac{\sigma(x) = (C, \&l) \quad h(l) = (C, \overline{fv}) \quad fv(f) = (T,v)}{(h;\sigma;x.f) \rightsquigarrow (T,v)}$$

Figure 5: Operational semantics for expressions.

$\boxed{h;\ \sigma \vdash v : T}$

$$\text{(R-C)}\quad \frac{}{h;\ \sigma \vdash c : \mathit{Bool}}$$

$$\text{(R-NONUNIQUE)}\quad \frac{\text{ClassDecl} \in ct \quad h(l) = (\rho\ C, \overline{fv}) \quad \rho \neq \mathit{Unique}}{h;\ \sigma \vdash \&l : \rho\ C}$$

$$\text{(R-NONUNIQUE)}\quad \frac{\begin{array}{c}\text{ClassDecl} \in ct \quad h(l) = (\mathit{Unique}\ C, \overline{fv}) \\ (\exists! x, \sigma(x) = (\mathit{Unique}\ C, \&l)) \vee \neg(\exists x, \sigma(x) = (\mathit{Unique}\ C, \&l))\end{array}}{h;\ \sigma \vdash \&l : \mathit{Unique}\ C}$$

Figure 6: Runtime value type.

$\boxed{(h; \sigma; S) \rightsquigarrow (h'; \sigma')}$

$$\text{(S-SKIP)}\quad \frac{}{(h; \sigma; \mathit{skip}) \rightsquigarrow (h;\ \sigma)}$$

$$\text{(S-ASSIGN-C) (OMIT LATER)}\quad \frac{(h; \sigma; y) \rightsquigarrow (\mathit{TBool}, v) \quad \sigma(x) \neq \mathit{None}}{(h; \sigma; x := y) \rightsquigarrow (h;\ \sigma[x \mapsto (\mathit{TBool}, v)])}$$

$$\text{(S-ASSIGN-S)}\quad \frac{(h; \sigma; y) \rightsquigarrow (\mathit{Shared}\ C, v) \quad \sigma(x) \neq \mathit{None}}{(h; \sigma; x := y) \rightsquigarrow (h;\ \sigma[x \mapsto (\mathit{Shared}\ C, v)])}$$

$$\text{(S-ASSIGN-U)}\quad \frac{(h; \sigma; y) \rightsquigarrow (\mathit{Unique}\ C, v) \quad \sigma(x) \neq \mathit{None}}{(h; \sigma; x := y) \rightsquigarrow (h;\ \sigma[y \mapsto (\perp C, v)][x \mapsto (\mathit{Shared}\ C, v)])}$$

$$\text{(S-FACC)}\quad \frac{h;\ \sigma \vdash y.f \rightsquigarrow (T, v) \quad \sigma(x) \neq \mathit{None}}{(h; \sigma; x := y.f) \rightsquigarrow (h;\ \sigma[x \mapsto (T, v)])}$$

$$\text{(S-FUPD-S)}\quad \frac{\begin{array}{c}(h; \sigma; y) \rightsquigarrow (\mathit{Shared}\ C, \&o) \quad (h; \sigma; x) \rightsquigarrow (\mathit{Shared}\ C', \&l) \\ h(l) = (\mathit{Shared}\ C', \overline{fv}) \quad fv(f) = (\mathit{Shared}\ C, v')\end{array}}{(h; \sigma; x.f := y) \rightsquigarrow (h[l \mapsto (\mathit{Shared}\ C', \overline{fv}[f \mapsto (\mathit{Shared}\ C, v)])]);\ \sigma)}$$

$$\text{(S-FUPD-U)}\quad \frac{\begin{array}{c}(h; \sigma; y) \rightsquigarrow (\mathit{Unique}\ C, \&o) \quad h(o) = (\mathit{Unique}\ C, \overline{fv'}) \quad (h; \sigma; x) \rightsquigarrow (\mathit{Shared}\ C', \&l) \\ h(l) = (\mathit{Shared}\ C', \overline{fv}) \quad fv(f) = (\mathit{Shared}\ C, v')\end{array}}{(h; \sigma; x.f := y) \rightsquigarrow (h[o \mapsto (\mathit{Shared}\ C, \overline{fv'})][l \mapsto (\mathit{Shared}\ C', \overline{fv}[f \mapsto (T, v)])]);\ \sigma[y \mapsto (\perp C)])}$$

Figure 7: Operational semantics for statements (1).

$\boxed{(h;\sigma;S) \rightsquigarrow (h';\sigma')}$

(S-MCall)
$$\frac{\begin{array}{c}\sigma(y) = (C_i, \&l) \quad \text{ClassDecl}_i = \text{Class } C_i\{\ \overline{F_n}\ K\ \overline{M}\} \\ M(m) = \text{def } m(\text{this} : C,\ \overline{p:T}) : \text{ret} : T_r : \{S\} \\ \sigma(x) \neq None \quad \sigma(z) \neq None \\ (h;\sigma;S[y/this, \overline{z}/\overline{p}]) \rightsquigarrow (h',\sigma') \quad (h';\sigma';ret) \rightsquigarrow (T_r, v_r)\end{array}}{(h;\sigma;x := y.m(\overline{z})) \rightsquigarrow (h';\ \sigma'[x \mapsto (T_r, v_r)])}$$

(S-let-S)
$$\frac{\begin{array}{c}(h;\ \sigma;e) \rightsquigarrow (Shared\ C, \&l) \quad FreeVar^r(S,\sigma) = x \\ (h;\sigma, x \mapsto (Shared\ C, \&l); S) \rightsquigarrow (h';\ \sigma', x \mapsto (\rho\ C, r))\end{array}}{(h;\sigma;var\ x : T := e\ in\ S) \rightsquigarrow (h';\ \sigma')}$$

(S-let-U)
$$\frac{\begin{array}{c}(h;\ \sigma;y) \rightsquigarrow (Unique\ C, \&l) \quad FreeVar^r(S,\sigma) = x \\ (h;\sigma[y \mapsto (\perp\ C, \&l)], x \mapsto (Unique\ C, \&l); S) \rightsquigarrow (h';\ \sigma', x \mapsto (\rho\ C, r))\end{array}}{(h;\sigma;var\ x : T := y\ in\ S) \rightsquigarrow (h';\ \sigma')}$$

(S-letnew)
$$\frac{\begin{array}{c}(h;\ \sigma;\overline{y}) \rightsquigarrow \overline{(T,v)}) \quad FreeVar^r(S,\sigma) = x \quad l\ is\ fresh \\ \text{ClassDecl} = \text{Class } C\{\ \overline{F}\ K\ \overline{M}\} \quad K = C(\overline{p:T_f}) : ret : \rho\ C : \{...\} \\ (h, l \mapsto (\rho\ C, \overline{(T,v)}); \sigma, x \mapsto (\rho\ C, \&l), S) \rightsquigarrow (h';\sigma', x \mapsto (\rho'\ C, v))\end{array}}{(h;\sigma;var\ x : \rho\ C := new\ \rho\ C(\overline{y})\ in\ S) \rightsquigarrow (h';\ \sigma')}$$

(S-iftrue)
$$\frac{\begin{array}{c}(h;\ \sigma;e) \rightsquigarrow (Bool, True) \\ (h;\sigma;S_1) \rightsquigarrow (h';\ \sigma') \\ FreeVar^r(S_1,\sigma) = \emptyset\end{array}}{(h;\sigma;if\ e\ then\ S_1\ else\ S_2) \rightsquigarrow (h';\ \sigma')}$$

(S-iffalse)
$$\frac{\begin{array}{c}(h;\ \sigma;e) \rightsquigarrow (Bool, False) \\ (h;\sigma;S_2) \rightsquigarrow (h';\ \sigma') \\ FreeVar^r(S_2,\sigma) = \emptyset\end{array}}{(h;\sigma;if\ e\ then\ S_1\ else\ S_2) \rightsquigarrow (h';\ \sigma')}$$

(S-looptrue)
$$\frac{\begin{array}{c}(h;\ \sigma;e) \rightsquigarrow (Bool, True) \\ (h;\sigma;S^c) \rightsquigarrow (h';\ \sigma')\end{array}}{(h;\sigma;while\ e\ do\ S\ in\ c) \rightsquigarrow (h'';\ \sigma'')}$$

(S-loopfalse)
$$\frac{(h;\ \sigma;e) \rightsquigarrow (Bool, False)}{(h;\sigma;while\ e\ do\ S\ in\ c) \rightsquigarrow (h;\ \sigma)}$$

(S-sequence)
$$\frac{(h;\sigma;S_1) \rightsquigarrow (h';\ \sigma') \quad (h';\sigma';S_2) \rightsquigarrow (h'';\ \sigma'')}{(h;\sigma;S_1;S_2) \rightsquigarrow (h'';\ \sigma'')}$$

Figure 8: Operational semantics for statements (2).

$\boxed{\mathit{StoreOK}\ \Gamma\ \sigma\ h\ ct}$

$$\frac{\begin{array}{c} dom(\Gamma) = dom(\sigma) \quad \mathrm{WF}(ct) \\ \forall x \in dom(\Gamma),\ \Gamma \vdash x : T \implies (\exists C\ v, \sigma(x) = (\perp\ C, v)) \vee \\ (\exists v, \sigma(x) = (T, v) \wedge \Gamma \vdash x : T \wedge h;\ \sigma \vdash v : T) \end{array}}{\mathit{StoreOK}\ \Gamma\ \sigma\ h\ ct}$$

$\boxed{\mathit{HeapOK}\ \Gamma\ \sigma\ h\ ct}$

$$\frac{\begin{array}{c} \forall o \in dom(h), h(o) = (\rho\ C, \overline{fv}) \implies \\ \text{ClassDecl} = \text{Class } C\{\overline{F}\ K\ \overline{M}\} \wedge length(\overline{fs}) = length(\overline{F}) \wedge \\ (\forall f \in [0, length(\overline{F})), \overline{F}(f) = T_f \wedge \overline{fv}(f) = (T_f, v) \wedge h; \sigma \vdash v : T_f \end{array}}{\mathit{HeapOK}\ \Gamma\ \sigma\ h\ ct}$$

$\boxed{\mathit{HeapStoreOK}\ \Gamma\ \sigma\ h\ ct}$

$$\frac{\begin{array}{c} \forall x \in dom(\Gamma), \Gamma \vdash x : \rho\ C \implies \exists l\ \overline{fv}, \sigma(x) = (C, \&l) \wedge h(l) = (C, \overline{fv}) \wedge \\ \text{ClassDecl} = \text{Class } C\{\overline{F}\ K\ \overline{M}\} \wedge (\forall f, F(f) = T_f \implies \\ (\exists v, \overline{fv}(f) = (T_f, v) \wedge \Gamma \vdash x.f : T_f \wedge h; \sigma \vdash v : T \end{array}}{\mathit{HeapStoreOK}\ \Gamma\ \sigma\ h\ ct}$$

$\boxed{\mathit{CtxOK}\ \Gamma\ \sigma\ h\ ct}$

$$\frac{\mathit{StoreOK}\ \Gamma\ \sigma\ h\ ct \wedge \mathit{HeapOK}\ \Gamma\ \sigma\ h\ ct \wedge \mathit{HeapStoreOK}\ \Gamma\ \sigma\ h\ ct}{\mathit{CtxOK}\ \Gamma\ \sigma\ h\ ct}$$

Figure 9: Definition of safety properties.

$\boxed{\text{Type Safety}}$

$$\mathit{CtxOK}\ \Gamma\ \sigma\ h\ ct \wedge \Gamma \vdash S \dashv \Gamma' \implies \exists \sigma'\ h', (h;\ \sigma, S) \rightsquigarrow (h';\ \sigma') \wedge \mathit{CtxOK}\ \Gamma'\ \sigma'\ h'\ ct$$

Figure 10: Type safety theorem (progress).