1 Definitions

P is the set of program variables used in rust program. Common names a, b, c

L is the set of logic variables used in refinement types. Common names: α, β

$\Gamma = (\mu, \sigma)$ is a tuple containing a function $\mu : P \to L$ mapping all program variables to their (current) logic variable and a set of formulas σ over L. During execution of statements, the set increases monotonically

τ is a user defined type $\{\alpha : b \mid \varphi\}$. Where α is a logic variable from L, b is a base type from Rust (like `i32`) and φ is a formula over variables in L.

2 Typing Rules

Abbreviations We write:

- Γ, c for $(\mu, \sigma \wedge c)$
- $\Gamma[a \mapsto \alpha]$ for $(\mu[a \mapsto \alpha], \sigma)$

2.1 Expression Typing: $\Gamma \vdash e : \tau$

$$\text{LIT}\ \frac{l \text{ fresh} \qquad \text{base\_ty}(v) = b}{\Gamma \vdash v : \{l : b \mid l \doteq v\} \Rightarrow \Gamma}$$

$$\text{VAR}\ \frac{\beta \text{ fresh} \qquad \mu(x) = \beta}{\Gamma = (\mu, \sigma) \vdash x : \{\alpha : b \mid \beta \doteq \alpha\} \Rightarrow \Gamma}$$

$$\text{VAR-REF}\ \frac{\Gamma \vdash y : \tau \qquad \Gamma \vdash x : \{\beta : \&b \mid \beta \doteq \&y\}}{\Gamma \vdash *x : \tau \Rightarrow \Gamma}$$

$$\text{IF}\ \frac{\Gamma \vdash c : \text{bool} \Rightarrow \Gamma_c \qquad \Gamma_c, c \vdash c_t : \tau \Rightarrow \Gamma_t \qquad \Gamma_c, \neg c \vdash c_e : \tau \Rightarrow \Gamma_t}{\Gamma \vdash \text{if } c \text{ then } c_t \text{ else } c_e : \tau \Rightarrow \Gamma_t}$$

$$\text{WHILE}\ \frac{\Gamma_I, c \vdash s \Rightarrow \Gamma_I' \qquad \Gamma_I' \preceq \Gamma_I}{\Gamma_I \vdash \texttt{while}(c)s \Rightarrow \Gamma_I, \neg c}$$

$$\text{SEQ}\ \frac{\Gamma \vdash s_1 : \tau_1 \Rightarrow \Gamma' \qquad \Gamma' \vdash \bar{s} : \tau \Rightarrow \Gamma''}{\Gamma \vdash s_1; \bar{s} : \tau \Rightarrow \Gamma''}$$

$$\text{ADD}\ \frac{\Gamma \vdash e_1 : \{v_1 : b \mid \varphi_1\} \Rightarrow \Gamma' \qquad \Gamma' \vdash e_2 : \{v_2 : b \mid \varphi_2\} \Rightarrow \Gamma''}{\Gamma \vdash e_1 + e_2 : \{v : b \mid v \doteq [e_1] + [e_2]\} \Rightarrow \Gamma'', \varphi_1, \varphi_2}$$

$$\text{ASSIGN}\ \frac{\Gamma \vdash e : \{\beta \mid \varphi\} \Rightarrow \Gamma'}{\Gamma \vdash x = e : \tau \Rightarrow \Gamma'[x \mapsto \beta], \varphi}$$

$$\text{ASSIGN-STRONG}\ \frac{\Gamma \vdash e : \{\beta \mid \varphi\} \Rightarrow \Gamma' \qquad \Gamma \vdash x : \{\alpha : \&b \mid \alpha \doteq \&y\}}{\Gamma \vdash *x = e : \tau \Rightarrow \Gamma'[y \mapsto \beta], \varphi}$$

$$\text{ASSIGN-WEAK}\ \frac{\Gamma \vdash e : \tau \Rightarrow \Gamma' \qquad \Gamma \vdash x : \{\alpha : \&b \mid \alpha \doteq \&y \lor \alpha \doteq \&z \lor \dots\} \qquad \Gamma \vdash \tau \preceq \tau_y \qquad \Gamma \vdash y : \tau_y}{\Gamma \vdash *x = e : \tau \Rightarrow \Gamma'}$$

$$\text{FN-CALL}\ \frac{(\{a \mapsto \tau_a, \dots\}, \{\alpha \doteq \mu(a), \dots, \varphi_\alpha, \dots\}) \preceq \Gamma \qquad f : (\{\alpha \mid \varphi_\alpha\} \Rightarrow \{\alpha' \mid \varphi'_\alpha\}, \dots)}{\Gamma \vdash f(a, \dots) \Rightarrow (\mu[a \mapsto \alpha', \dots], \sigma \land \varphi'_\alpha \land \dots)}$$

$$\text{INTRO-SUB}\ \frac{\Gamma \vdash e : \tau \qquad \Gamma \vdash \tau \preceq \tau'}{\Gamma \vdash e\ \texttt{as}\ \tau' : \tau'}$$

2.2 Sub-Typing Rules: $\Gamma \vdash \tau \preceq \tau'$

$$\preceq\text{-TY}\ \frac{\sigma \land \varphi'[\beta \triangleright \alpha] \vDash \varphi}{\Gamma = (\mu, \sigma) \vdash \{\alpha \mid \varphi\} \preceq \{\beta \mid \varphi'\}}$$

alternative (should be equivalent):

$$\preceq\text{-TY-ALT}\ \frac{\Gamma[f \mapsto \alpha], \varphi \preceq \Gamma[f \mapsto \beta], \varphi' \qquad f \text{ fresh}}{\Gamma \vdash \{\alpha \mid \varphi\} \preceq \{\beta \mid \varphi'\}}$$

2.3 Sub-Context Rules: $\Gamma \preceq \Gamma'$

$$\preceq\text{-CTX}\ \frac{\sigma'[\mu(\alpha) \triangleright \mu'(\alpha) \ \mid \alpha \in dom(\mu)] \vDash \sigma \qquad dom(\mu) \subseteq dom(\mu')}{(\sigma, \mu) \preceq (\sigma', \mu')}$$