A Polymoprhic Typing Rules

Polymorphic constant signatures:

$\mathsf{sig}(n) = \texttt{int}$ n an integer constant $\quad \mathsf{sig}(\oplus) = \texttt{int} \to \texttt{int} \to \texttt{int}$ for $\oplus \in \{+, -, *, \% \ldots\}$
$\mathsf{sig}(\texttt{true}) = \texttt{bool} \quad \mathsf{sig}(\texttt{false}) = \texttt{bool}$
$\mathsf{sig}(\sim) = \forall\alpha.\, \alpha \to \alpha \to \texttt{bool}$ for $\sim \in \{<, >, =, \leq, \geq\}$
$\mathsf{sig}(\texttt{[ ]}) = \forall\alpha.\, \alpha\texttt{list} \quad \mathsf{sig}((::)) = \forall\alpha.\, \alpha \to \alpha\ \texttt{list} \to \alpha\ \texttt{list}$
$\mathsf{sig}((\ ,\ )) = \forall\alpha\beta.\, \alpha \to \beta \to \alpha * \beta$

Constants:

$$\frac{}{\Gamma \vdash\ c\ :\ \tau'}\ \textsc{Const}$$

where c is a constant listed above, $\mathsf{sig}(c) = \forall\alpha_1 \ldots \alpha_n.\, \tau$ and there exist $\sigma_1, \ldots, \sigma_n$ such that $\tau' = \tau[\sigma_1/\alpha_1; \ldots; \sigma_n/\alpha_n]$

Variables:

$$\frac{}{\Gamma \vdash x : \tau'}\ \textsc{Var}$$

where $\forall\alpha_1 \ldots \alpha_n.\, \tau = \Gamma(x)$ and there exist $\sigma_1, \ldots, \sigma_n$ such that $\tau' = \tau[\sigma_1/\alpha_1; \ldots; \sigma_n/\alpha_n]$

Connectives:

$$\frac{\Gamma \vdash e_1 : \texttt{bool} \qquad \Gamma \vdash e_2 : \texttt{bool}}{\Gamma \vdash e_1\ \&\&\ e_2 : \texttt{bool}}\ \textsc{Conn} \qquad \frac{\Gamma \vdash e_1 : \texttt{bool} \qquad \Gamma \vdash e_2 : \texttt{bool}}{\Gamma \vdash e_1\ ||\ e_2 : \texttt{bool}}\ \textsc{Conn}$$

If_then_else rule:

$$\frac{\Gamma \vdash e_c : \texttt{bool} \qquad \Gamma \vdash e_t : \tau \qquad \Gamma \vdash e_e : \tau}{\Gamma \vdash \texttt{if}\ e_c\ \texttt{then}\ e_t\ \texttt{else}\ e_e : \tau}\ \textsc{If}$$

Application rule:

$$\frac{\Gamma \vdash e_1 : \tau_1 \to \tau_2 \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1\ e_2 : \tau_2}\ \textsc{App}$$

Function rule:

$$\frac{[x : \tau_1] + \Gamma \vdash e : \tau_2}{\Gamma \vdash \texttt{fun}\ x\ \texttt{->}\ e : \tau_1 \to \tau_2}\ \textsc{Fun}$$

Let rule:

$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad [x : \mathsf{Gen}(\tau_1, \Gamma)] + \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash \texttt{let}\ x\ \texttt{=}\ e_1\ \texttt{in}\ e_2 : \tau_2}\ \textsc{Let}$$

Let Rec rule:

$$\frac{[x : \tau_1] + \Gamma \vdash e_1 : \tau_1 \qquad [x : \mathsf{Gen}(\tau_1, \Gamma)] + \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash \texttt{let rec}\ x\ \texttt{=}\ e_1\ \texttt{in}\ e_2 : \tau_2}\ \textsc{Rec}$$

B Floyd-Hoare Logic

Simple Imperative Programming Language:

$$C ::= I\texttt{:=}E \mid C\ \texttt{;}\ C \mid \texttt{if}\ B\ \texttt{then}\ C\ \texttt{else}\ C\ \texttt{fi} \mid \texttt{while}\ B\ \texttt{od}\ C\ \texttt{od}\ C$$

where I ranges over program identifiers, E ranges over program arithmetic expressions, and B ranges boolean-valued expressions.

Rules:

Assignment Axiom:

$$\frac{}{\{P[e/x]\}; x\texttt{:=}e\{P\}}$$

Sequencing Rule:

$$\frac{\{P\}C_1\{Q\} \quad \{Q\}C_2\{R\}}{\{P\}C_1\ \texttt{;}\ C_2\{R\}}$$

If_then_else Rule:

$$\frac{\{P \text{ and } B\}C_1\{Q\} \quad \{P \text{ and (not } B)\}C_2\{Q\}}{\{P\}\texttt{if}\ B\ \texttt{then}\ C_1\ \texttt{else}\ C_2\ \texttt{fi}\{Q\}}$$

While Rule:

$$\frac{\{P \text{ and } B\}C\{Q\}}{\{P\}\texttt{while}\ B\ \texttt{do}\ C\ \texttt{od}\{P \text{ and (not } B)\}}$$

Precondition Strengthening:

$$\frac{P \Longrightarrow P' \quad \{P'\}C\{Q\}}{\{P\}C\{Q\}}$$

Postcondition Weakening

$$\frac{\{P\}C\{Q'\} \quad Q' \Longrightarrow Q}{\{P\}C\{Q\}}$$