Basic Types	b	$::=$	$\alpha \mid x : \tau \rightarrow \tau \mid C\,\bar{\tau}\,\bar{r} \mid \tau\,\tau$
Types	τ	$::=$	$\{v : b \mid r\} \mid Cl\,\bar{\tau}$
Abstract Refinements	π	$::=$	$\forall \langle p{:}\tau \rangle . \pi \mid \tau$
Type Schemata	σ	$::=$	$\forall \alpha . \sigma \mid \pi$
Refinements	r	$::=$	(ar, cr)
Abstract Refinements	ar	$::=$	$[] \mid p\,\bar{e}, ar$
Concrete Refinements	cr	$::=$	$k\,[e/x] \mid pr \mid cr \wedge cr$
Predicates	pr	$::=$	$true \mid false \mid \bigwedge \bar{pr} \mid \bigvee \bar{pr} \mid \neg pr \mid pr \Rightarrow pr \mid pr \Leftrightarrow pr$
		$\mid$	$e \mid e\,[= \mid \neq \mid > \mid < \mid \geq \mid \leq]\,e$
Expressions	e	$::=$	$c \mid n \mid x \mid c\,\bar{e} \mid if\ pr\ then\ e\ else\ e \mid e\,[+ \mid - \mid * \mid / \mid \%]\,e$

Figure 1: Syntax of `liquidHaskell`

NV:♣ TODO :

1. Add termination checker.♣

Constraint Generation (without the termination checker)

Type synthesis $\quad \Gamma \vdash e \uparrow \sigma ; C$

$$\frac{(x, \{v : b \mid e\}) \in \Gamma}{\Gamma \vdash x \uparrow \{v : b \mid e \wedge x = v\}; \emptyset} \qquad \frac{(x, \sigma \in \Gamma) \qquad \sigma \neq \{v : b \mid e\}}{\Gamma \vdash x \uparrow \sigma ; \emptyset}$$

$$\frac{}{\Gamma \vdash c \uparrow \{v : ty(c) \mid v = c\}; \emptyset}$$

$$\frac{\Gamma \vdash e \uparrow \forall \alpha . \sigma ; C \qquad \tau' = if\ isGeneric(\alpha, \sigma)\ then\ freshTy(\tau)\ else\ trueTy(\tau)}{\Gamma \vdash e\,[\tau] \uparrow \sigma\,[\tau'/\alpha]\,; (\mathtt{WfC}\,\Gamma\,\tau', C)}$$

$$\frac{\Gamma \vdash e_1 \uparrow \tau_1 ; C_1 \qquad \Gamma \vdash e_2 \downarrow \tau_x ; C_2 \qquad (\tau_1', C_p) = freshPreds(\Gamma, \tau_1) \qquad x : \tau_x \rightarrow \tau = \tau_1'}{\Gamma \vdash e_1\ e_2 \uparrow \tau\,[e_2/x]\,; (C_1, C_2, C_p)}$$

$$\frac{\Gamma \vdash e \uparrow \sigma ; C}{\Gamma \vdash [\Lambda \alpha]\,e \uparrow \forall \alpha . \sigma ; C}$$

$$\frac{\Gamma, x : \tau_x \vdash e \uparrow \tau ; C \qquad t_x = freshty(varType\ x)}{\Gamma \vdash (\lambda x . e) \uparrow (x : \tau_x \rightarrow \tau) ; (\mathtt{WfC}\,\Gamma\,\tau_x, C)}$$

$$\frac{\Gamma \vdash e \uparrow \sigma ; C}{\Gamma \vdash Tick\ t\ e \uparrow \sigma ; C}$$

$$\frac{}{\Gamma \vdash Cast\ e\ c \uparrow trueTyExpr(Cast\ e\ c) ; \emptyset}$$

$$\frac{}{\Gamma \vdash \mathit{Coercion}\ c \uparrow \mathit{trueTyExpr}(\mathit{Coercion}\ c); \emptyset}$$

$$\frac{\sigma = \mathit{freshTy}(e) \qquad \Gamma \vdash \texttt{let}\ x_i = e_{x_i}\ \texttt{in}\ e \downarrow \sigma; C}{\Gamma \vdash \texttt{let}\ x_i = e_{x_i}\ \texttt{in}\ e \uparrow \sigma; (\texttt{WfC}\ \Gamma\ \sigma, C)}$$

$$\frac{\sigma = \mathit{freshTy}(e) \qquad \Gamma \vdash \mathit{Case}\ e\ x\ \mathit{alt}_i \downarrow \sigma; C}{\Gamma \vdash \mathit{Case}\ e\ x\ \mathit{alt}_i \uparrow \sigma; (\texttt{WfC}\ \Gamma\ \sigma, C)}$$

Type checking $\Gamma \vdash e \downarrow \sigma; C$

$$\frac{\tau_x = \mathit{userTypes}(x) \qquad \Gamma \vdash e_x \downarrow \tau_x; C_x \qquad \Gamma, x{:}\tau_x \vdash e \downarrow \sigma; C}{\Gamma \vdash \texttt{let}\ x = e_x\ \texttt{in}\ e \downarrow \sigma; (C_x, C)}$$

$$\frac{x \notin \mathit{userTypes} \qquad \Gamma \vdash e_x \uparrow \tau_x; C_x \qquad \Gamma, x{:}\tau_x \vdash e \downarrow \sigma; C}{\Gamma \vdash \texttt{let}\ x = e_x\ \texttt{in}\ e \downarrow \sigma; (C_x, C)}$$

$$\frac{\Gamma, \overline{x_i : \tau_{x_i}} \vdash e \downarrow \sigma; C \qquad (C_{x_i}, \tau_{x_i}) = \mathit{varTemplate}(x_i) \qquad \Gamma, \overline{x_i : \tau_{x_i}} \vdash e_{x_i} \downarrow \tau_{x_i}; C'_{x_i}}{\Gamma \vdash \texttt{let}\ x_i = e_{x_i}\ \texttt{in}\ e \uparrow \sigma; (C_{x_i}, C'_{x_i}, C)}$$

$$\frac{\Gamma \vdash e \uparrow \tau_x; C_x \qquad (\Gamma, x : \tau_x); x \downarrow \mathit{alt}_i{:}\sigma; C_i}{\Gamma \vdash \mathit{Case}\ e\ x\ \mathit{alt}_i \downarrow \sigma; (C_x, C_i)}$$

$$\frac{\Gamma \vdash e \downarrow \sigma\,[\alpha/\alpha']\,; C}{\Gamma \vdash [\Lambda\alpha]\, e \downarrow \forall\alpha'.\sigma; C}$$

$$\frac{\Gamma, x : \tau_y \vdash e \downarrow \tau\,[x/y]\,; C}{\Gamma \vdash \lambda x.e \downarrow (y : \tau_y \to \tau); C}$$

$$\frac{\Gamma \vdash e \downarrow \sigma; C}{\Gamma \vdash \mathit{Tick}\ t\ e \downarrow \sigma; C}$$

$$\frac{\sigma' = \mathit{trueTy}(\mathit{Cast}\ c\ e)}{\Gamma \vdash \mathit{Cast}\ c\ e \downarrow \sigma; (C, \texttt{SubC}\ \Gamma\ \sigma'\ \sigma)}$$

$$\frac{\Gamma, p{:}\tau \vdash e \downarrow \sigma; C}{\Gamma \vdash e \downarrow \forall\,\langle p{:}\tau\rangle\,.\sigma; C}$$

$$\frac{\Gamma \vdash e \uparrow \sigma'; C \qquad (\sigma'', C_p) = \mathit{freshPreds}(\Gamma, \sigma')}{\Gamma \vdash e \downarrow \sigma; (C, C_p, \texttt{SubC}\ \Gamma\ \sigma''\ \sigma)}$$

$\Gamma; x \downarrow \mathit{alt}{:}\sigma; C$

$$\frac{(x, \tau_x^0) \in \Gamma \qquad \tau_x^0 = \{v : C\ t_{Cl}\ r_{Cj} \mid r\} \qquad ty(C) = \forall\alpha_l p_j.y_1{:}t_1 \to \ldots y_n{:}t_n \to t \qquad \theta = [t_{Cl}/\alpha_l]\,\left[r_{Cj}/p_j\right]\,[x_i/y_i] \qquad \tau_{x_i} = \theta t_i \qquad \tau_x = \theta t \wedge \tau_x^0 \wedge \mathit{dataConTy}(C, x_i) \qquad \Gamma, x{:}\tau_x, x_i{:}\tau_{x_i} \vdash e \downarrow \sigma; C}{\Gamma; x \downarrow (C, x_i, e){:}\sigma; C}$$

Helper Functions

$isGeneric(\alpha, \sigma)$ – not constrained by class predicates

$$isGeneric(\alpha, \sigma) \Leftrightarrow \alpha \notin ClassConstraints(\sigma)$$

$$\begin{aligned} classConstraints(\forall \alpha.\sigma) &= classConstraints(\sigma) \\ classConstraints(\forall p.\sigma) &= classConstraints(\sigma) \\ classConstraints(C \alpha_i \to \tau) &= \alpha_i \cup classConstraints(\tau) \end{aligned}$$

$freshTy(\sigma)$ – type with liquid variables for all refinements

$$\begin{array}{lcl} freshTy(\{v : \alpha \mid r\}) & = & \{v : \alpha \mid fref\} \\ freshTy(\{v : x : \tau_x \to \tau \mid r\}) & = & \{v : x : freshTy(\tau_x) \to freshTy(\tau) \mid tref\} \\ freshTy(\{v : C\, \overline{\tau}\, \overline{r} \mid r\}) & = & \{v : C\, \overline{freshTy(\tau)}\, \overline{fref} \mid fref\} \\ freshTy(\{v : \tau_1\, \tau_2 \mid r\}) & = & \{v : freshTy(\tau_1)\, freshTy(\tau_2) \mid tref\} \\ freshTy(Cl\, \overline{\tau}) & = & Cl\, \overline{\tau} \\ freshTy(\forall \alpha.\sigma) & = & \forall \alpha.freshTy(\sigma) \\ freshTy(\forall \langle p{:}\tau \rangle .\sigma) & = & \forall \langle p{:}\tau \rangle .freshTy(\sigma) \end{array}$$

where $fref = ([], k_i), tref = ([], true)$

$trueTy(\sigma)$ – type with true for all refinements

$$\begin{array}{lcl} trueTy(\{v : \alpha \mid r\}) & = & \{v : \alpha \mid tref\} \\ trueTy(\{v : x : \tau_x \to \tau \mid r\}) & = & \{v : x : trueTy(\tau_x) \to trueTy(\tau) \mid tref\} \\ trueTy(\{v : C\, \overline{\tau}\, \overline{r} \mid r\}) & = & \{v : C\, \overline{trueTy(\tau)}\, \overline{tref} \mid tref\} \\ trueTy(\{v : \tau_1\, \tau_2 \mid r\}) & = & \{v : trueTy(\tau_1)\, trueTy(\tau_2) \mid tref\} \\ trueTy(Cl\, \overline{\tau}) & = & Cl\, \overline{\tau} \\ trueTy(\forall \alpha.\sigma) & = & \forall \alpha.trueTy(\sigma) \\ trueTy(\forall \langle p{:}\tau \rangle .\sigma) & = & \forall \langle p{:}\tau \rangle .trueTy(\sigma) \end{array}$$

$freshPreds(\Gamma, \sigma)$ – replace predicate occurrences with liquid variables

$$\begin{array}{lcll} freshPreds(\Gamma, \forall \alpha.\sigma) & = & (\forall \alpha.\sigma', C) & where \quad (\sigma', C) = freshPreds(\Gamma, \sigma) \\ freshPreds(\Gamma, \forall \langle p{:}\tau \rangle .\sigma) & = & (\sigma' [k_i/p], (C, \mathtt{WfC}\ \Gamma'\ k_i)) & where \quad (\sigma', C) = freshPreds(\Gamma, \sigma) \\ & & & \qquad x_1{:}\tau_1 \to \ldots x_n{:}\tau_n \to Prop = \tau \\ & & & \qquad \Gamma' = \Gamma, x_1{:}\tau_1 \to \ldots x_{n-1}{:}\tau_{n-1} \\ freshPreds(\Gamma, \tau) & = & (\tau, \emptyset) & \end{array}$$

$trueTyExpr(e)$ – type of expression with true for all refinements

$varTemplate(x)$ – type for variable x, user specified type or a fresh type

$$dataConTy(C, x_i) = \begin{cases} Prop\ v & C = True \\ \neg(Prop\ v) & C = False \\ v = x_1 & C = I\# \\ v = Cx_i & \text{otherwise} \end{cases}$$