1 Language

$$
\begin{array}{rcll}
x, D & ::= & \mathit{String} & \text{(Variables, Data)} \\
e & ::= & x \mid f(\overline{e}) \mid D(\overline{e}) & \text{(Expressions)} \\
 & & \mid \texttt{BAD} \mid \texttt{UNR} & \\
p & ::= & \text{let } f\ \overline{x} = \text{case } e \text{ of } [(p_1 \to e_1)\dots(p_n \to e_n)] ::: t, p & \text{(Programs)} \\
 & & \mid \text{let } f\ \overline{x} = e ::: t, p \mid \epsilon & \\
\Gamma & ::= & (f \mapsto t), \Gamma \mid (f\ \vec{x} \mapsto e), \Gamma \mid \epsilon & \text{(Contexts)} \\
t & ::= & \{x \mid e\} \mid x : t \to t \mid \texttt{Any} & \text{(Contracts)} \\
\phi & ::= & \top \mid \bot \mid x \mid \neg\phi \mid \phi \wedge \phi \mid \forall x, \phi & \text{(FOF)}
\end{array}
$$

With the constraint that the Haskell expression in the contract must return a Bool.
$\vec{x}$ denotes a vector of variables and Γ a context and multiset (which enables multiple contracts.)
$t \geq t' \iff \{e|x \in t'\} \subset \{e|x \in t\}$ (t is more restrictive than t')
case-expression can't be nested (just as lambdas) which is simpler for the FOL translation.

2 Encoding

2.1 Expressions

2.1.1 Axioms

- $\|x\| = x$
- $\|f(\overline{e})\| = f(\|\overline{e}\|)$
- $\|D(\overline{e})\| = D(\|\overline{e}\|)$
- $\|\texttt{BAD}\| = \texttt{BAD}_\phi$ (maybe we must specify that $\texttt{BAD}_\phi$ is not $\top$ or $\bot$)
- $\|\texttt{UNR}\| = \texttt{UNR}_\phi$ (same here)

2.1.2 Contract satisfaction

- $[\![\Gamma \vdash \texttt{BAD} \in t]\!] = \bot$
- $[\![\Gamma \vdash \texttt{UNR} \in t]\!] = \top$
- $$\begin{array}{rcll} [\![\Gamma \vdash x \in t]\!] & = & \top & \text{if } x \mapsto t \in \Gamma \\ & & \bot & \text{otherwise.} \end{array}$$
- $$\begin{array}{rcl} [\![\Gamma \vdash f\overline{(e_1, \dots, e_n)} ::: t_1 \to \dots \to t_n \to t_c \in t_c]\!] & = & \bigwedge_{1 \leq i \leq n} [\![\Gamma \vdash e_i \in t_i]\!] \\ [\![\Gamma \vdash f\overline{(e_1, \dots, e_n)} ::: t \in t']\!] & = & \bot \text{ ie type mismatch.} \end{array}$$
- $[\![\Gamma \vdash D(\overline{e}) \in t]\!] = (\|\overline{e}\| \neq \texttt{BAD}_\phi \leftrightarrow \|D(\overline{e})\| \neq \texttt{BAD}_\phi) \wedge$ check D as a function application

Invariant: $[\![\Gamma \vdash e \in t]\!] = \phi \iff (\phi = \top \iff \Gamma, e \text{ satisfies } t)$

2.2 Programs

Assume $t = t_1 \to \cdots \to t_n \to t_c$ and $\overline{x} = (x_1, \ldots, x_n)$

- $[\![\Gamma \vdash \text{let } f(\overline{x}) = \text{case } e \text{ of } [(p_1 \to e_1), \ldots, (p_n \to e_n)] \in t]\!] = \|e\| \neq \mathtt{BAD}_\phi \wedge (\forall \vec{z}\ p_i[\vec{z}] = e \to [\![\Gamma, \vec{z} \mapsto \mathtt{Ok} \vdash e_i \in t]\!] \vee \|e'\| = \mathtt{UNR}_\phi)$
  ($\vec{z}$ is a short hand for $(z_1, \ldots, z_k)$ and e' is the whole expression)
- $$\begin{aligned} [\![\Gamma \vdash \text{ let } f(\overline{x}) = e ::: t, p]\!] &= \forall x_1, \ldots, x_n, \{x_1 \in t_1\}, \ldots, \{x_n \in t_n\} \\ & \quad [\![\Gamma, x_1 \mapsto t_1, \ldots, x_n \mapsto t_n, f^\star \mapsto t \vdash e[f^\star/f] \in t_c]\!] \\ & \quad \wedge [\![\Gamma, f \mapsto t \vdash p]\!] \\ \{x \in t\} &= t(x) \wedge \forall y, t(y) \iff ([\![\Gamma \vdash p(y) \in \{z | z = \mathtt{True}\}]\!]) \end{aligned}$$

($t = \{y | p(y)\}$ and True refers to the Haskell boolean value (and not $\top$))