## 1 Language

$$\begin{array}{rcll}
x, D & ::= & \mathit{String} & \text{(Variables, Data)} \\
e & ::= & x \mid e\ e \mid D\ e \mid \texttt{BAD} \mid \texttt{UNR} & \text{(Expressions)} \\
p & ::= & \text{let } f\ x_1, \ldots, x_n = \text{case } e \text{ of } [(p_1 \to e_1) \ldots (p_n \to e_n)] ::: t, p & \text{(Programs)} \\
 & & \mid \text{let } f\ x_1, \ldots, x_n = e ::: t, p \mid \epsilon & \\
t & ::= & \{x \mid e\} \mid x : t \to t \mid \texttt{Any} & \text{(Contracts)} \\
\phi & ::= & \top \mid \bot \mid x \mid \neg\phi \mid \phi \wedge \phi \mid \forall x, \phi & \text{(FOF)}
\end{array}$$

With the constraint that the Haskell expression in the contract must return a Bool.
$t \geq t' \iff \{e|x \in t'\} \subset \{e|x \in t\}$ ($t$ 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

- $[\![e \in \{x|p(x)\}]\!] = p(e)$
- $[\![e \in t_1 \to t_2]\!] = \forall x, [\![x \in t_1]\!] \to [\![e\ x \in t_2]\!]$

### 2.2 Programs

Assume $t = t_1 \to \cdots \to t_n \to t_c$.

- $[\![\text{let } f\ x_1 \ldots x_n = e ::: t]\!] = (\forall_{1 \leq i \leq n} x_i, \bigwedge_{1 \leq i \leq n} [\![x_i \in t_i]\!]\ \forall f^\star [\![f^\star \in t]\!]) \to [\![e[f^\star/f] \in t_c]\!]$
- $[\![\text{let } f\ x_1 \ldots x_n = \text{ case } e \text{ of } [(p_i, e_i)] \in t]\!] = (\forall x_i, \bigwedge_{i \leq n} [\![x_i \in t_i]\!]\ \forall f^\star [\![f^\star \in t]\!]) \to (\bigwedge_{j \leq k} p_j = e \to [\![e_j[f^\star/f] \in t_c]\!])$