1 Grammars

1.1 λ-lifted haskell subset

$$\begin{array}{rcll}
u, e & := & x \mid f \mid e\ e \mid D \mid \texttt{BAD} \mid n & \text{(Expression)} \\
p & := & \Delta_1 \ldots \Delta_n & \text{(Program)} \\
\Delta & := & d \mid \mathit{transp}(d, c) \mid \mathit{opaque}(d, c) & \text{(Defintion and contract)} \\
d & := & f\ x_1 \ldots x_n = e \mid f\ x_1 \ldots x_n = \text{ case } e \text{ of } [(\mathit{pat}_i, e_i)] & \text{(Definition)} \\
\mathit{pat} & := & D\ x_1 \ldots x_n & \text{(Pattern)}
\end{array}$$

1.2 FOL

$$\begin{array}{rcll}
t & := & x \mid \mathrm{app}(t_1, t_2) \mid k & \text{(Term)} \\
k & := & D \mid f \mid ? \mid \texttt{BAD} \mid \texttt{UNR} & \text{(Constant)} \\
\phi & := & \forall x.\phi \mid \phi \to \phi \mid \neg\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \mathit{true} \mid t = t \mid \mathrm{CF}(t) & \text{(Formula)}
\end{array}$$

1.3 Contracts

$$\begin{array}{rcl}
c & := & x : c_1 \to c_2 \\
 & \mid & (c_1, c_2) \\
 & \mid & \{x \mid e\} \\
 & \mid & \texttt{Any}
\end{array}$$

We can consider CF as a user contract.

2 Translation

We define several translations: $\mathcal{E}[\![]\!], \mathcal{D}[\![]\!], \mathcal{S}[\![]\!], [\![]\!]$.

$$\begin{array}{rcl}
\mathcal{E}[\![]\!] & :: & \mathit{Expression} \to \mathit{Term} \\
\mathcal{D}[\![]\!] & :: & \mathit{Definition} \to \mathit{FOF} \\
\mathcal{S}[\![]\!] & :: & \mathit{Expression} \to \mathit{Contract} \to \mathit{FOF} \\
[\![]\!] & :: & \mathit{Definition} \to \mathit{Contract} \to \mathit{FOF}
\end{array}$$

2.1 $\mathcal{E}[\![]\!]$

$\mathcal{E}[\![e]\!]$ is a term. The translation is direct.

2.2 $\mathcal{D}[\![]\!]$

$\mathcal{D}[\![d]\!]$ is a first-order formula.

$$\begin{array}{rcl}
\mathcal{D}[\![f\ x_1 \ldots x_n = e]\!] & = & \forall x_1 \ldots x_n . \mathcal{E}[\![f\ x_1 \ldots x_n]\!] = \mathcal{E}[\![e]\!] \\
\mathcal{D}[\![f\ x_1 \ldots x_n = \text{case } e \text{ of } [D_i\ \overline{z} \mapsto e_i]]\!] & = & \forall x_1 \ldots x_n\ (\bigwedge_i (\forall \overline{z}\ \mathcal{E}[\![e]\!] = \mathcal{E}[\![D_i\ \overline{z}]\!] \rightarrow \mathcal{E}[\![f\ x_1 \ldots x_n]\!] = \mathcal{E}[\![e_i]\!]) \\
 & & \wedge \mathcal{E}[\![e]\!] = \mathtt{BAD} \rightarrow \mathcal{E}[\![f\ x_1 \ldots x_n]\!] = \mathtt{BAD}) \\
 & & \wedge [\![f\ x_1 \ldots x_n]\!] = \mathtt{UNR} \bigvee_i (\mathtt{HD}(e) = D_i) \qquad (\varphi)
\end{array}$$

2.3 $\mathcal{S}[\![]\!]$

$\mathcal{S}[\![e \in c]\!]$ is a first-order formula.

$$\begin{array}{rcll}
\mathcal{S}[\![e \in \mathtt{Any}]\!] & = & \mathit{true} & (1) \\
\mathcal{S}[\![e \in \{x \mid u\}]\!] & = & \mathtt{UNR} \vee (\mathrm{CF}(\mathcal{E}[\![e]\!]) \wedge \mathrm{CF}(\mathcal{E}[\![u[e/x]]\!] \neq \mathtt{BAD}) \wedge \mathcal{E}[\![u[e/x]]\!] \neq \mathit{False}) & (2) \\
\mathcal{S}[\![e \in x : c_1 \rightarrow c_2]\!] & = & \forall x_1 . \mathcal{S}[\![x_1 \in c_1]\!] \rightarrow \mathcal{S}[\![e\ x_1 \in c_2[x_1/x]]\!] & (3)
\end{array}$$

(4)

False is a data constructor here.

Remark: we follow the semantics of the POPL paper but it's a bit restrictive. e.g. in equation 2 we could use the alternate semantics (namely B1 in the POPL paper) :

$$\mathcal{S}[\![e \in \{x \mid u\}]\!] = \mathtt{UNR} \vee (\mathcal{E}[\![u[e/x]]\!] \neq \mathtt{BAD} \wedge \mathcal{E}[\![u[e/x]]\!] \neq \mathit{False})$$

2.4 $[\![]\!]$

It's the final translation, which takes a function definition and its contract and returns a first-order formula

$$\begin{array}{rcl}
[\![\mathit{opaque}(f\ x_1 \ldots x_n = e, c)]\!] & = & \mathcal{D}[\![f\ x_1 \ldots x_n = e[f_p/f]]\!] \wedge \mathcal{S}[\![f \in c]\!] \wedge \mathcal{S}[\![f_p \in c]\!] \\
[\![\mathit{transp}(f\ x_1 \ldots x_n = e, c)]\!] & = & \mathcal{D}[\![f\ x_1 \ldots x_n = e[f_p/f]]\!] \wedge \mathcal{S}[\![f \in c]\!] \wedge \mathcal{S}[\![f_p \in c]\!] \\
[\![f\ x_1 \ldots x_n = e]\!] & = & [\![\mathit{opaque}(f\ x_1 \ldots x_n) = e, \mathit{Ok} \rightarrow \mathit{Ok})]\!]
\end{array}$$

We'd like a typical contract-checking session to go like this:

1. Start with an empty theory T.
2. Let $f\ x_1 \ldots x_n = e \in c$ be an opaque function definition to check wrt contract c. Check (with equinox) the consistency of the theory $T' = T \cup [\![f\ x_1 \ldots x_n = e \in c]\!]$
3. If T' is consistent then let $T = \mathcal{S}[\![f \in c]\!] \cup T$ and go to 2. with the next function definition; otherwise give a counter-example and ask the user for refinement of the contracts and/or lemmas(?)

3 Questions

Here are some open issues, design choices and equinox-related questions.

1. Nested implications may lead to existential quantification, is it troublesome in equinox? (altough we don't have this case here)
2. More generally, does equinox accept any FOF the grammar here defines?
3. In the section $\mathcal{D}[\![]\!]$, we believe replacing φ by $\vee[\![f x_1 \dots x_n]\!] = \texttt{UNR}$ is equivalent. What's better for equinox?
4. Is the session stuff realistic? It looks like it can have a quadratic behaviour but maybe with the right API it's ok?