Implementing Observational Type theory in Epigram 2.

Wouter Swierstra (with a lot of help from my friends)
slippery tar-pit

two notions of equality

\[ \equiv \quad \text{for typechecking} \]

\[ \equiv \quad \text{for reasoning, traditionally} \]

(same for set)

\[ \Gamma + A \text{ set } \Gamma + a : A \quad \Gamma + a' : A \]

\[ \Gamma + a =_A a' \text{ set} \]

\[ \Gamma + a : A \]

\[ \Gamma - a^A : a =_A a \]

and now the schism

extensional TT

\[ \Gamma + q : a = a' \]

\[ \Gamma - a : a = a' : A \]

intensional TT

\[ \Gamma ; x : A ; q : a = a x \vdash P[x : q] \]

\[ \Gamma - a : A \quad \Gamma + p : P[a; a^A] \]

\[ \Gamma + r : a = a' \]

\[ \Gamma + r (\text{subst}^a_{x : q} P[x : q] \mid p) : P[a ; r] \]

with \( \overline{a}^A (\text{subst}^a_{x : q} P[x : q] \mid p) \rightarrow p \)
extensional ETT is jolly useful:

\( (\Pi x : S. f x = g x) \rightarrow f = g \)

but undecidable

(checking \(\equiv\) involves guessing \(=\))

In ETT terms are no longer evidence,
Epigram is a language of evidence

- No termination checker, but explicit structural recursion ("rec").

- No coverage checker, but explicit pattern matching ("case").

- Adding the axiom of extensionality is not in the spirit of the language.
Things get even worse....

Extensional type theory has a clear underlying intuition:
What can you do with functions, but apply them?

But what about:

\[
\begin{align*}
\text{data} & \quad f : N \to N \\
\text{isId } f : \star
\end{align*}
\]

where

\[
\text{idIsId : isId } (\lambda n \to n)
\]

Use functions to index data types.
So what?

Clearly:
\[ \text{id} \circ \text{id} : \text{id} \circ \text{id} \ (\lambda \ n \rightarrow n) \]

But we can prove:
\[ (\lambda \ n \rightarrow n) = (\lambda \ n \rightarrow n + 0) \]

And use substitution to make an inhabitant of
\[ \text{id} \circ \text{id} \ (\lambda \ n \rightarrow n + 0) \]
in the empty context.

But what constructor made this term?
Story so far:

We have to be very, very, very careful if we want some form of extensional equality for Epigram:

- we want more evidence

- extensionality and indexed data families are tricky. Just ask Peter!
What is Observational type theory?

- Thorsten Altenkirch, LICS ’99
  Extension type theory in intensional type theory. LICS ’99

- Thorsten Altenkirch and Conor McBride,
  Towards Observational type theory. 2006.

For one thing, it’s not my idea.
Remember \textit{id}?
Substitution by non-refl proofs is hairy.  
\hspace{1cm} (start with refl, add ext.)

\textbf{Observational type theory} is backwards:

1) \textbf{what is observable?}  
\hspace{1cm} (ext.)

2) What about refl?
Hello John Major!

Start from heterogeneous equality:

\[
\begin{align*}
S_0 \quad \text{set} \quad & \quad S_1 \quad \text{set} \\
\hline
S_0 &= S_1 \quad \text{set}
\end{align*}
\]

\[
\begin{align*}
s_0 : S_0 \quad & \quad s_1 : S_1 \\
(\underbrace{s_0 : S_0}_{\text{set}}) &= (\underbrace{s_1 : S_1}_{\text{set}})
\end{align*}
\]

so far so good.
Now proceed structurally over types thinking about observable behaviour.

\[ P : (\Pi x : S_0. T_0. [x]) = (\Pi x : S. T. [x]) \]

project \( P : S_0 = S \),
Similarly,

\[ P : \Pi \times : S_0 . T_0 [\times] = \Pi \times : S . T \times [\times] ; \]

\[ p : (s_0 : S_0 = s_1 : S_1) \]

\[ \text{apply } P \ p : T_0 [s_0] = T_1 [s_1] \]

And similarly for \( \Sigma \).

But what about \textit{extensionality}?
The conversion rule converts definitionally equal terms.

In QTT we build an explicit coercion between provably equal terms.

\[
\frac{P : S_0 = S_1 \quad s_0 : S_0}{\text{coerce} \quad P \ s_0 : S_1}
\]
Even better: coercion does not change terms. We can show that:

\[ P : S_0 = S_1 \; ; \; s_0 : S_0 \]  

\[ \text{coherence} \; P \; s_0 : (s_0 : S_0) = (\text{coerce } P \; s_0) \]
Sceptic:
"I don't trust you!"

Me:
"and I don't trust myself!"
1000 lines of Agda on a single slide:

- construct a universe $U$ closed under $\Pi$, $\Sigma$, $\nu$ with $\emptyset$, 1, 2.
- define equality on $U$.
- define equality on $\text{el}(U)$.
- prove that these equalities are symmetric and transitive.
- define $\text{coerce} : (u_1 : U) \to (u_2 : U) \to (u_1 \equiv u_2) \to \text{el } u_1 \to \text{el } u_2$.
- prove $\text{coherence} : (u_1 : U) \to (u_2 : U) \to (P : u_1 \equiv u_2) \to (x : \text{el } u_1) \to x \equiv \text{coerce } u_1, u_2 P x$. 
Sceptic:
But you have not proven refl and you use it!

Me:
The meta-theory of Agda says refl is
Ok - see the paper.
Besides, if I could prove refl, OTT would
be little more than a construction in Agda!
Proof irrelevance

Coercion never inspects the proof:

- kills off the off-diagonal cases
- projects out parts for recursive calls.

That's all (Agda keeps me honest).
What can we do with Observational type theory?

- derive extensionality!
- prove the induction principle for Not — and add even more evidence. No more awkward schemas!
- opens the door for type theoretic treatment of quotients, coinduction, ....