The Joy of QIITs

QIITs = Quotient Inductive Inductive Types

Thorsten Altenkirch
jww Paolo Capriotti, Ambrus Kaposi, Andras Kovac,
Nicolai Kraus, Jakob von Raumer

Functional Programming Laboratory
School of Computer Science
University of Nottingham

April 6, 2018

What is equality?

- Intensional Type Theory:
  - Types are defined by their elements.
  - Equality type reflects judgmental equality.
  - Lack of extensionality.
- Homotopy Type Theory
  - Types are defined by elements and equalities.
  - Equality types provide access to the equality structure.
  - Very extensional

Higher Inductive Types

- Inductive Types:
  - Value constructors: to construct elements
  - Equality constructors: to construct equalities
- Applications:
  - Synthetic homotopy theory:
    - ⋆ Definition of the circle (S^1)
    - ⋆ Higher spheres (S^n)
    - ⋆ Torus, ...
  - Set level structures:
    - ⋆ Cauchy Reals
    - ⋆ Partiality Monad
    - ⋆ Intrinsic Type Theory

Quotient Inductive Inductive Types

- Set level HITs : Quotient Inductive Types (QITs)
- HITs with set or propositional truncation constructors.
- Inductive-inductive types (IITs): define inductive types that depend on each other, e.g.

$$A : \mathbf{Set}, B : A \to \mathbf{Set}$$

- QITs + IITs = QIITS.
- All the set level examples are QIITs.

The Cauchy Reals

- Standard definition: quotient converging sequences. Identify converging sequences whose difference converges to 0.
- Try to show that the reals are Cauchy complete: every converging sequence of reals has a limit.
- To show this we need to commute quotients and functions.
- In general this is equivalent to the axiom of choice.
- HoTT book: Define the reals as the Cauchy completion of the rationals.
- This requires a QIIT since we define a equivalence relation at the same time.
- We avoid using axiom of choice!
- Assuming AC the two definitions are equivalent.

The Partiality Monad

- Given A : **Set** define $A_\perp$: *Set* the set of partial computations over A.
- Old approach:
  - coinductively define delayed computation as the terminal coalgebra of $F\,X = A + X$.
  - Quotient delayed computation by those that differ only by a finite delay.
- We would like to show:
  - $(-)_\perp$ is a monad (computational effect).
  - $A_\perp$ is an ω-CPO (has directed least upper bounds and a least element).
- It seems that we need AC again.

The Partiality Monad as a QIIT

- Instead we define $A_\perp$ as the (underlying set) of the free ω-CPO.
- This is a ω-CPO by definition.
- It is a monad by abstract nonsense (composition of a free and forgetful functor).
- We are using a QIIT to define the elements, the order relation and the equality at the same time.
- Assuming AC we can show that this definition is equivalent to the previous one.

A simplified example: permutable trees

- Given $A : \mathbf{Set}$ let us define the type of permutable trees $T(A) : \mathbf{Set}$ of A-branching trees where we identify trees upto permutation of subtrees.
- Define A-branching trees $T_0(A) : \mathbf{Set}$ inductively

$$\begin{aligned} \mathrm{leaf} &: T_0(A) \\ \mathrm{node} &: (A \to T_0(A)) \to T_0(A) \end{aligned}$$

- We define the permutability relation

$$\_ \sim \_ : T_0(A) \to T_0(A) \to \mathbf{Prop}$$

inductively by

$$\begin{aligned} \mathrm{perm} &: \forall \pi : A = A.\mathrm{node}(f \circ \pi) \sim \mathrm{node}(f) \\ \mathrm{leaf}^{=} &: \mathrm{leaf} \sim \mathrm{leaf} \\ \mathrm{node}^{=} &: (\forall x : A.\mathrm{node}(f(x)) \sim \mathrm{node}(g(x))) \to \mathrm{node}(f) \sim \mathrm{node}(g) \end{aligned}$$

A simplified example: permutable trees

- Define $T(A) = T_0(A)/\sim$.
- Can we derive

$$\overline{\mathrm{node}} : (A \to T(A)) \to T(A)$$

  such that

$$[\mathrm{node}(f)] = \overline{\mathrm{node}}([\_] \circ f)$$

- Using AC

$$\begin{aligned} [\mathrm{node}(f)] &= \overline{\mathrm{node}}[f]_{A\to\sim} && (AC) \\ &= \overline{\mathrm{node}}([\_] \circ f) && (\mathrm{node}^{=}) \end{aligned}$$

- Derive $\overline{\mathrm{node}}$ using surjectivity of $[\_]$ and unique choice.

Permutable trees as a QIT

- Define permutable A-branching trees $T(A)$: **Set** inductively

$$
\begin{aligned}
\mathrm{leaf} &: T(A) \\
\mathrm{node} &: (A \to T(A)) \to T(A) \\
\mathrm{perm} &: \forall \pi : A = A . \mathrm{node}(f \circ \pi) = \mathrm{node}(f)
\end{aligned}
$$

- Using AC we can show that this is equivalent to the quotient definition.
- However, we can completely avoid AC in this development.

QITs to avoid choice

- QI(I)Ts are a constructive principle that enables us to avoid unnecessary uses of AC.
- Indeed, Lumsdaine and Shulman given an example of a QIT that cannot be derived without AC.

Semantics of higher inductive types

Peter LeFanu Lumsdaine, Mike Shulman. 2017.
arXiv:1705.07088 [math.LO]

Type Theory in Type Theory as a QIIT

$$
\begin{aligned}
\mathrm{Con} &: \mathbf{Set} \\
\mathrm{Ty} &: \mathrm{Con} \to \mathit{Set} \\
\mathrm{Tm} &: \Pi\Gamma : \mathrm{Con}.\mathrm{Ty}(\Gamma) \to \mathbf{Set} \\
\mathrm{Tms} &: \mathrm{Con} \to \mathrm{Con} \to \mathbf{Set} \\
&\vdots \\
\mathrm{Pi} &: \Pi A : \mathrm{Ty}(\Gamma), B : \mathrm{Ty}(\Gamma.A).\mathrm{Ty}(\Gamma) \\
&\vdots \\
\mathrm{lam} &: \mathrm{Tm}(\Gamma.A, B) \to \mathrm{Tm}(\Gamma, \mathrm{Pi}(A, B)) \\
\mathrm{app} &: \mathrm{Tm}(\Gamma, \mathrm{Pi}(A, B)) \to \mathrm{Tm}(\Gamma.A, B) \\
&\vdots \\
\beta &: \Pi t : \mathrm{Tm}(\Gamma.A, B).\mathrm{lam}(\mathrm{app}(f)) = f
\end{aligned}
$$

Type Theory in Type Theory

- Intrinsic syntax (no preterms).
- This is the initial category with families by definition.
- Could be defined as a quotient type since there are no higher constructors.
- However, this is very laborious!
- Since the QIIT is truncated we cannot define the standard model in **Set**.

Type Theory in Type Theory using Quotient Inductive Types

Thorsten Altenkirch and Ambrus Kaposi. 2016. POPL.

Codes for QIITs

- Kaposi and Kovac propose to use TTinTT to specify QIITs.
- They restrict Π-types so that we can only define 1st order functions:

$$\begin{aligned} \mathrm{U} &: \mathrm{Ty}(\Gamma) \\ \mathrm{El} &: \mathrm{Tm}(\Gamma, U) \to \mathrm{Ty}(\Gamma) \\ \mathrm{Pi}^r &: \Pi A : \mathrm{Tm}(\Gamma, \mathrm{U}), B : \mathrm{Ty}(\Gamma.\mathrm{El}(A)).\mathrm{Ty}(\Gamma) \end{aligned}$$

- To allow non-recursive arguments they add a higher (and large): constructor:

$$\mathrm{Pi}^n : \Pi A : \mathbf{Set} B : A \to \mathrm{Ty}(\Gamma).\mathrm{Ty}(\Gamma)$$

- They also add equality types.
- We only need application for both Pi-types and transport for equality.
- Contexts in this type theory correspond to QIITs, e.g. natural numbers

$$\Gamma_N = N : \mathrm{U}, z : \mathrm{El}(N), s : \mathrm{Pi}^r(\mathrm{N}, \mathrm{El}(\mathrm{N}))$$

A universal QIIT?

- We can define an interpretation of this type theory that assigns categories to contexts,
- We can also derive an object in this category which is the term algebra.
- Conjecture: this is the initial object.
- Conjecture: The QIIT defining this type theory is universal (all other QIITs can be derived from it).

Going further

- Is there a simple universal QIIT?
- Can we define a higher syntax of Type Theory as a HIIT (and define the set-model)?
- What are applications of proper HITs (and HIITs) outside of synthetic homotopy theory?