From iterated parametricity to indexed semi-simplicial and semi-cubical sets: a formal construction

Hugo Herbelin

joint work with Ramkumar Ramachandra

TYPES 2022
Unary and binary parametricity

For a given typed language, parametricity associates to each type an observational characterisation of the behaviour of its inhabitants. E.g.:

**Unary parametricity** (a form of realisability)

To \( A : \text{Type} \), associate \( A^\ast : A \to \text{Type} \)

Example: \( f \in (A \to B) \) iff for all \( x \in A \), we have \( f x \in B \)

**Binary parametricity** (a form of bisimilarity)

To \( A : \text{Type} \), associate \( A^\ast : A \times A \to \text{Type} \)

Example: \( f =_{A \to B} g \) iff for all \( a \) and \( b \) such that \( a =_{A} b \), we have \( f a =_{B} g b \)
Iterated parametricity

Parametricity can be iterated:

E.g. in the binary case, associate to $A$:

$$A_\ast : A \times A \to \text{Type},$$

$$A_{\ast\ast} : \begin{array}{c}
  a : A \\
  b : A
\end{array} \frac{r:A_\ast(a,c)\quad q:A_\ast(c,d)}{c : A} \frac{p:A_\ast(a,b)}{d : A} \frac{s:A_\ast(b,d)}{\to \text{Type}}$$

... giving rise to a semi-cubical set structure (binary case) or augmented semi-simplicial set structure (unary case, as communicated to us by Moeneclaey).
Fibered vs indexed representation of dependent types

There are two standard ways to represent a dependent type over a type $B$:

- the dependent "indexed" way:
  \[ P : B \to \text{Type} \]

- the fibered way (as in "fibrational" categorical models):
  \[(A, f) : \Sigma A : \text{Type}. (A \to B)\]

A fundamental correspondence: $B \to \text{Type} \simeq \Sigma A : \text{Type}. (A \to B)$

Proof relies on univalence (or even without if $\simeq$ is defined in an enough extensional way):

\[
\begin{align*}
P & \quad \text{total space of } P \\
\lambda b : B. \Sigma a : A. (fa = b) & \quad \text{fiber of } b \\
(A, f) & \quad \text{(A, } f) \\
\end{align*}
\]
Fibered vs indexed parametricity

Consequently, two equivalent characterisations of the assignment to a type of an iterated family of (relevant) predicates/relations over this type:

\[
\begin{align*}
\text{iterated fibered} & \quad \text{iterated indexed} \\
\vdots & \quad \vdots \\
A_{**} : \text{Type} & \quad A_{**} : \Pi ab : A, A_*(a, b) \to \\
 & \quad \Pi cd : A. A_*(c, d) \to \\
 & \quad A_*(a, c) \times A_*(b, d) \to \text{Type} \\
\downarrow \downarrow \downarrow \downarrow & \quad A_* : A \times A \to \text{Type} \\
A_* : \text{Type} & \quad A_* : \text{Type} \\
\downarrow & \\
A : \text{Type} & \\
\end{align*}
\]

+ coherence conditions

For connections between the two approaches, see e.g. Atkey-Johann-Ghani 2014, Bernardy-Coquand-Moulin 2014, Altenkirch-Kaposi 2015, Tabareau-Tanter-Sozeau 2018, ...
What was done?

The full axiom-free formalisation of the type of iterated parametricity sets of arity $\nu$ in indexed form in Coq:

$$\nu\text{-parametric-type} \triangleq \Sigma A : \text{hSet}_l. (\Sigma A_* : ((\nu \to A) \to \text{hSet}_l). (\Sigma A_{**} : \ldots \ldots))$$

as a dependent (coinductive) stream of higher-order relations (see abstract for details), up to a requirement of functional extensionality over functions of domain $\nu$ (sources at github.com/artagnon/bonak).

Related formalisations: Sozeau-Tabareau’s groupoid model 2013, H.’s indexed semi-simplicial types 2013, Tabareau-Tanter-Sozeau’s 2-groupoid model 2013, Altenkirch-Boulier-Kaposi-Tabareau’s setoid model 2019, Finster-Allioux-Sozeau’s opetopic types 2021, Chen-Kraus’ semi-simplicial types in CwF 2021, ...
Technical issues I

Strongly dependent construction where:

- level $n$ requires defining a notion of border at level $n - 1$
- which depends on defining a notion of border restriction map from level $n - 1$
- which depends on a coherence condition between restrictions from levels $n - 1$ and $n - 2$

We did not succeed to get full well-founded induction (unmanageable dependencies).

- Instead, we worked on blocks of three levels $n - 2$, $n - 1$, $n$ at once, that we stepwise slid
Technical issues II

Strongly dependent construction, proofs of $p \leq q$ occur in statements with antagonistic requirements:

- ability to do case analysis on inequality proofs
- convenience of definitional proof irrelevance on inequality proofs

For that purpose, we switch between two representations shown equivalent:

- a definitionally proof-relevant inductive formulation in Type
- a definitionally proof-irrelevant definition obtained by a Yoneda construction (i.e. $n \leq_Y p \triangleq \Pi q. q \leq n \rightarrow q \leq p$) on top of a recursive definition of inequality in strict Prop.
An algebra of inequality proofs

In particular, we use three instances of the following algebra of inequality proofs:

\[
\begin{align*}
\text{contra} &: \quad 0 = n + 1 \quad \rightarrow \quad \text{False} \\
\text{init} &: \quad 0 \leq n \\
\Diamond &: \quad n \leq n \\
\uparrow &: \quad n \leq m \land m \leq p \quad \rightarrow \quad n \leq p \\
\uparrow &: \quad n \leq m \quad \rightarrow \quad n \leq m + 1 \\
\downarrow &: \quad n + 1 \leq m \quad \rightarrow \quad n \leq m \\
\downarrow &: \quad n + 1 \leq m + 1 \quad \rightarrow \quad n \leq m \\
\uparrow &: \quad n \leq m \quad \rightarrow \quad n + 1 \leq m + 1
\end{align*}
\]

Incidentally suggests to canonically infer inequality proofs using type classes (but not attempted)
Conclusions and open questions

- A first formal rather “canonical” reusable step in direction of developing (possibly univalent) parametricity models of type theory within ETT (in line with Altenkirch-Kaposi 2014, Altenkirch-Kaposi-Shulman 2022), as an alternative to the popular approach of providing presheaf (i.e. fibered) models in ZF.

- Our kind of proofs is still a challenge for proof assistants. For example:
  - how to get definitionally that a proof of equality of pairs is a pair of proofs of equalities? (a “new generation” type theory)
  - can we justify new definitional equalities e.g. those built by induction from steps preserving definitional equalities (as in the proof of $x + 0 = x$)?
  - what kind of automatic normalisation for combinations of equality proofs? (can we learn from the HoTT libraries?)