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 : **Type**, associate $A_\star : A \to \textsf{Type}$

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 : **Type**, associate $A_\star : A \times A \to \textsf{Type}$

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:

$A_\star : A \times A \to \mathsf{Type}$,

$$A_{\star\star} : \begin{tikzcd} a : A \arrow[r, dash, "r : A_\star(a,c)"] \arrow[d, dash, "p : A_\star(a,b)"'] & c : A \arrow[d, dash, "q : A_\star(c,d)"] \\ b : A \arrow[r, dash, "s : A_\star(b,d)"] & d : A \end{tikzcd} \to \mathsf{Type}$$

...

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:

$$P : B \to \mathsf{Type}$$

- the fibered way (as in "fibrational" categorical models):

$$(A, f) : \Sigma A : \mathsf{Type}.\,(A \to B)$$

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

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

$$\begin{array}{lcl} P & \mapsto & (\overbrace{\Sigma b : B.\, P\, b}^{\text{total space of } P}, \pi_1) \\ \lambda b : B.\, \underbrace{\Sigma a : A.\,(f\,a = b)}_{\text{fiber of } b} & \leftarrow\!\shortmid & (A, f) \end{array}$$

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:

iterated fibered	*iterated indexed*
$\vdots$	$\vdots$
$A_{\star\star} : \mathsf{Type}$	$A_{\star\star} : \Pi ab : A, A_\star(a,b) \to \Pi cd : A.\ A_\star(c,d) \to A_\star(a,c) \times A_\star(b,d) \to \mathsf{Type}$
$\downdownarrows\downdownarrows$	
$A_\star : \mathsf{Type}$	$A_\star : A \times A \to \mathsf{Type}$
$\downdownarrows$	
$A : \mathsf{Type}$	$A : \mathsf{Type}$

$+$ *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 ν in indexed form in Coq:

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

as a dependent (coinductive) stream of higher-order relations (see abstract for details), up to a requirement of functional extensionality over functions of domain ν (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. $n \leq_{\mathcal{Y}} p \triangleq \Pi q.\, q \leq n \rightarrow q \leq p$) 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:

$$
\begin{array}{lclcl}
\mathsf{contra} & : & 0 = n + 1 & \rightarrow & \mathsf{False} \\
\mathsf{init} & : & & & 0 \leq n \\
\diamond & : & & & n \leq n \\
\updownarrow & : & n \leq m \wedge m \leq p & \rightarrow & n \leq p \\
\uparrow & : & n \leq m & \rightarrow & n \leq m + 1 \\
\downarrow & : & n + 1 \leq m & \rightarrow & n \leq m \\
\Downarrow & : & n + 1 \leq m + 1 & \rightarrow & n \leq m \\
\Uparrow & : & n \leq m & \rightarrow & n + 1 \leq m + 1
\end{array}
$$

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?)