Taking "algebraically" seriously in the definition of algebraically injective type

Martín Escardó

School of Computer Science,

University of Birmingham, UK

12th ASSUME seminar, 5th June 2025

Nottingham, UK

Abstract

Theorem. In a 1-topos, the following two categories are isomorphic, with an isomorphism that is the identity on objects:

1. Pullback-natural, associative, algebraically injective objects.

2. Algebras of the partial-map classifier (aka lifting) monad.

- Partial results towards the ∞-topos situation.
- We work in HoTT/UF.

I. Algebraic injectives (MSCS'2021)

Def. Algebraic injective structure on a type D consists of

1. An extension operation, for any types X and Y,

$$(-)|(-) : (X \to D) \times (X \hookrightarrow Y) \longrightarrow (Y \to D).$$

fibers are propositions.

2. For each map $f : X \to D$ and embedding $j : X \hookrightarrow Y$, a choice of an identification $(f|j) \circ j = f$, as illustrated by

$$\begin{tikzcd} X \arrow[rr, hook, "j"] \arrow[dr, "f"'] & & Y \arrow[dl, "f|j"] \\ & D & \end{tikzcd}$$

Some examples

MSCS'2021

(They need univalence)

1. $D := \mathcal{U}$

(a) $$\begin{tikzcd} X \arrow[rr, hook, "j"] \arrow[dr, "f"'] & & Y \arrow[dl] \\ & \mathcal{U} & \end{tikzcd}$$ $$(f \mid j)(y) := \prod_{(x,-) : \mathrm{fiber}\, j\, y} f(x).$$ (Right Kan extension.)

(b) Use Σ instead. (Left Kan extension.)

2. The type Ω of propositions. Use $\forall$ or $\exists$.

3. Universes of n-types.

4. Algebras of the lifting monad. We'll come back to this.

Homomorphisms of algebraic injectives

$$
\begin{tikzcd}
X \arrow[rr, hook, "j"] \arrow[dr, "f"'] \arrow[ddr, bend right] & & Y \arrow[dl, "f/j"] \arrow[ddl, bend left] \\
& D \arrow[d, "h"] & \\
& E &
\end{tikzcd}
$$

$$h \circ (f|j) = (h \circ f) | j$$

Pullback naturality

Previous examples are all pullback natural.

$$\begin{tikzcd}
A \arrow[r, hook, "k"] \arrow[d, "g"'] \arrow[dr, phantom, "\lrcorner"] & B \arrow[d, "h"] \arrow[ddl, bend left, "(f\circ g)|k"] \\
X \arrow[r, hook, "j"] \arrow[dr, "f"'] & Y \arrow[d, "f/j"] \\
& D
\end{tikzcd}$$

$$\boxed{(f|j)\circ h = (f\circ g)|k}$$

It is essential that the square is a pullback.

Consider the non-pullback square

$$\begin{tikzcd}
0 \arrow[r, hook] \arrow[d] & 1 \arrow[d] \\
1 \arrow[r, hook] & 1
\end{tikzcd}$$

for a counter-example.

Associativity

$$\begin{tikzcd} X \arrow[r, hook, "j"] \arrow[dr, "f"'] & Y \arrow[r, hook, "k"] \arrow[d, "f|j"] & Z \arrow[dl] \\ & D & \end{tikzcd} \qquad f|(k\circ j) = (f|j)|k$$

Examples (MSCS'2021) $D := \mathcal{U}$ with extension given by Π or Σ.

II. Algebraic flabby structure

MSCS'2021

$$\begin{tikzcd} P \arrow[r, hook] \arrow[dr, "f"'] & 1 \arrow[d, "\sqcup f"] \\ & D \end{tikzcd}$$

$\boxed{\text{Every partial element of } D \text{ can be extended to a total element}}$

N.B. T.F.A.E.

1. The map $P \to 1$ is an embedding.
2. The type P is a proposition.

Trivial fact. Algebraic injective structure is, in particular, algebraic flabby structure.

Homomorphisms

$$
\begin{tikzcd}
P \arrow[r, hook] \arrow[dr, "f"'] & 1 \arrow[d, "U_D f"] \arrow[dd, bend left=60, "U_E(h \circ f)"] \\
& D \arrow[d, "h"] \\
& E
\end{tikzcd}
$$

$$\boxed{h \circ U_D f = U_E (h \circ f)}$$

construction

Algebraic flabby → algebraic injective

MSCS'2021

$$\begin{tikzcd} X \arrow[r, hook, "j"] \arrow[dr, "f"'] & Y \arrow[d, "f|j"] \\ & D \end{tikzcd}$$

$$(f|j)(y) := \bigsqcup_{(x,-) : \underbrace{\mathrm{fiber}\ j\ y}_{P_y}} f\,x$$

Fact (new observation)

This algebraic injective structure is pullback natural.

(Not only fiber-natural)

$$\begin{tikzcd} P_y \arrow[r] \arrow[d, "pr_1"'] & 1 \arrow[d, "\bigsqcup(f \circ pr_1)"] \\ X \arrow[r, "f"'] & D \end{tikzcd}$$

Fiberwise extension.

$$\begin{tikzcd} P_y \arrow[r, hook] \arrow[d, "pr_1"'] \arrow[dr, phantom, "\lrcorner", very near start] & 1 \arrow[d, "y"] \arrow[ddl, bend left, "\bigsqcup(f \circ pr_1)"] \\ X \arrow[r, hook, "j"] \arrow[d, "f"'] & Y \arrow[dl, "f|j"] \\ D & \end{tikzcd}$$

Get this by flabbiness.

III. The lifting monad

M.H.E & C. Knapp CSL'2017 & TypeTopology 2018

$$\mathcal{L}X := \sum_{P:\Omega} (P \xrightarrow{\varphi} X)$$

The type of partial elements of X.

$$\begin{tikzcd}
X \arrow[d, mapsto] & X \arrow[r, "f"] \arrow[d, "\eta_X"'] & Y \arrow[d, "\eta_Y"] \\
(\mathbb{1}, \lambda - . x) & \mathcal{L}X \arrow[r, "\mathcal{L}f"] & \mathcal{L}Y \\
 & (P, \varphi) \arrow[r, mapsto] & (P, f \circ \varphi)
\end{tikzcd}$$

$$\text{is-def(ined)} : \mathcal{L}X \to \Omega := pr_1$$

$$\text{value} : (l : \mathcal{L}X) \to \text{is-def } l \to X := pr_2$$

$$\begin{tikzcd}
X \arrow[r, "g"] & \mathcal{L}Y \\
\mathcal{L}X \arrow[r, "g^{\#}"] & \mathcal{L}Y \\
(P, \varphi) \arrow[r, mapsto] & {}
\end{tikzcd}$$

$$\left(\left(\sum_{p:P} \text{is-def}(f(\varphi p))\right), \lambda (p, d) . \text{ value } (f(\varphi p))\, d \right)$$

is a prop, as required.

Monad algebras

1. Structure map

$$\bigsqcup : \mathcal{L} A \to A$$

Extend a partial element to a total element!
So $\mathcal{L}$-algebras give algebraic flabby structure.
(MSCS'2021)

2. Unit law

$$\bigsqcup (\lambda(-:\mathbb{1}).\, a) = a \quad \text{for every } a : A.$$

Extension "property", as for aflabby types.
(extension data!)

3. Associativity law

$$\bigsqcup_{p:P} \bigsqcup_{q:Q_p} f(p,q) = \bigsqcup_{r:\Sigma_{p:P} Q_q} f\, r$$

$P : \Omega$
$Q : P \to \Omega$
$f : \Sigma_{p:P} Q_p \to A$

Previously unaccounted for when discussing injectivity

IV. Putting I – III together

So $\mathcal{L}$-algebra structure = associative algebraic flabby structure.

Lemma Let $\sqcup$ be the algebraic flabby structure induced by a given algebraic injective structure $|$ that is pullback natural.

Then $\sqcup$ is associative iff $|$ is associative

$$\sqcup\sqcup = \sqcup$$

$$\begin{tikzcd} \bullet \arrow[r, hook] \arrow[dr] & \bullet \arrow[r, hook] \arrow[d] & \bullet \arrow[dl] \\ & \bullet & \end{tikzcd}$$

Ongoing work. Replace "iff" by "$\simeq$".
(For sets we have this.)

Lemma Let $|$ be the algebraic injective structure induced by a given algebraic flabby structure $\sqcup$.
Then $|$ is always pullback natural. (We've already discussed this.)

Lemma The round trip $\sqcup \longmapsto | \longmapsto \sqcup'$ is always the identity on both extension operators and extension data.

Lemma The round trip $| \longmapsto \sqcup \longmapsto |'$ is the identity on extension operators iff $|$ is pullback natural.

But what about <u>extension data</u>?

Ongoing.

Theorem. Let D be any type.

1. Then

pullback-natural, associative injective structure on D

$\iff$ associative algebraic flabby structure on D

$=$ $\mathcal{P}$-algebra structure on D.

2. If D is a set, then "$\iff$" in (1) becomes an equivalence "$\simeq$".

- what is missing to always have a type equivalence?
    - check that the pullback-naturality data is unchanged by round trips.
    - check that the associativity data is unchanged by round trips.

  (ongoing work, perhaps not difficult.)

- But there is still something else missing.

V. Is $\mathcal{L}$ really a monad?

Nobody knows what a monad on types is in HoTT/UF.

People do know what monads on ∞-toposes are, though.

But we don't know how to say that in the language of HoTT/UF.

The problem is how to specify coherence data for the monad laws.

Speculative ideas

This is the part of the talk in which I make a fool of myself.

$\boxed{\text{Try}}$ to define monad on types as follows.

1. Data

   (a) A function $T : \mathrm{Type} \to \mathrm{Type}$ (May well change type levels.)

   (b) A function $\eta : X \to TX$ (parametric in X.)

   (c) A function $(f \mapsto f^{\#}) : (X \to TY) \to (TX \to TY)$ (parametric in X, Y)

   Then we can define

   $$\mu : TTX \to TX := (\mathrm{id}_{TX})^{\#}$$

   $$T : (X \to Y) \to (TX \to TY) := f \mapsto (\eta_Y \circ f)^{\#}$$

2. No equations. Instead ...

2. Declare TX to be a free algebra w.r.t. TY's.

This is given by a universal property, and so is property rather than data.

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[dr, "\forall f"'] & TX \arrow[d, dashed, "\exists ! h"] & TTX \arrow[l, "\mu_X"'] \arrow[d, "TTh"] \\
& TY & TTY \arrow[l, "\mu_Y"]
\end{tikzcd}$$

The square says that h is an algebra homomorphism.

In HoTT/UF,

$$(f : X \longrightarrow TY) \to \text{is-contr}\left(\sum_{h : TX \to TY} (h \circ \eta_X \sim f) \times (h \circ \mu_X \sim \mu_Y \circ TTh)\right)$$

2.

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[dr, "\forall f"'] & TX \arrow[d, dashed, "\exists! h"] & TTX \arrow[l, "\mu_X"'] \arrow[d, "TTh"] \\
 & TY & TTY \arrow[l, "\mu_Y"]
\end{tikzcd}$$

In HoTT/UF,

$$(f : X \longrightarrow TY) \to \text{is-contr}\left(\sum_{h : TX \to TY} (h \circ \eta_X \sim f) \times (h \circ \mu_X \sim \mu_Y \circ TTh) \right)$$

Likely problem.

h should be $f^{\#}$, but how could we conclude that?

$\boxed{\text{The problem seems to disappear for idempotent monads}}$

Tentative definition. An idempotent monad on types consists of

1. A function $T : \mathrm{Type} \to \mathrm{Type}$.

   A family of functions $\eta_X : X \to TX$. data

   property

2. $$\begin{tikzcd} X \arrow[r, "\eta_X"] \arrow[dr, "\forall f"'] & TX \arrow[d, dashed, "\exists! h"] \\ & TY \end{tikzcd}$$

Here $\exists!$ is contractibility of Σ.

$\mathrm{ext} : (f : X \to TY) \to \exists!\, h : TX \to TY,\ h \circ \eta_X \sim f$.

Idea: In an idempotent monad, every map of free algebras is a homomorphism

Then define $f^{\#} := pr_1(\text{center of contraction}(\mathrm{ext}\, f))$.

$\nabla f := pr_2(\underline{\qquad\qquad\qquad\qquad})$.

That's the complete definition!

Then we can define $T : (X \to Y) \to TX \to TY$ and μ from $(-)^\#$ as before.

With Eric Finster we checked in my office whiteboard that the diagrams below commute.

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[dr, "f"'] & TX \arrow[d, "f^\#"] \arrow[dd, bend left=60, "(g^\# \circ f)^\#"] \\
Y \arrow[r] \arrow[dr, "g"'] & TY \arrow[d, "g^\#"] \\
& TZ
\end{tikzcd}$$

$$\begin{tikzcd}
T^3 \arrow[r, "T\mu"] \arrow[d, "\mu T"'] & T^2 \arrow[d, "\mu"] \\
T^2 \arrow[r, "\mu"'] & T
\end{tikzcd}
\qquad
\begin{tikzcd}
T \arrow[r, "\eta T"] \arrow[d, "T\eta"'] \arrow[dr, equal] & T^2 \arrow[d, "\mu"] \\
T^2 \arrow[r, "\mu"'] & T
\end{tikzcd}$$

$$\begin{tikzcd}
X \arrow[r, "\eta"] \arrow[dr, "\eta"'] & TX \arrow[d] \\
& TX
\end{tikzcd}
\qquad \eta^\# = id$$

Idempotency

The maps

$$TX \underset{\eta_{TX}}{\overset{T\eta_X}{\rightrightarrows}} TTX$$

are equal.

The first one is

$$(\eta_{TX} \circ \eta_X)^\#$$

$$X \xrightarrow{\eta_X} TX \xrightarrow{\eta_{TX}} TTX.$$

To show that they are equal, it is enough to show that

$$T\eta_X \circ \eta_X = (\eta_{TX} \circ \eta_X)^\# \circ \eta_X.$$

But this holds by the definition of $(-)^\#$ and by $\triangleleft$.

Summary of ∇ and questions

1. This idea seems to work for idempotent monads on types.
2. We only give the data $T : \mathrm{Type} \to \mathrm{Type}$ and $\eta_X : X \to TX$.
3. Then we give a property of these data, formulated as a universality condition.
4. From this we automatically get the usual equations for idempotent monads.
5. The equations should be automatically fully coherent, as they are deduced from property.
6. Can we do the same for arbitrary monads on types, following the above outline?