

Taking "algebraically" seriously
in the definition of
algebraically inductive 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 ,

$$(-) \mid (-) : (X \rightarrow D) \times (X \hookrightarrow Y) \rightarrow (Y \rightarrow D).$$

fibers are propositions.

2. For each map $f : X \rightarrow D$ and embedding $j : X \hookrightarrow Y$,

a choice of an identification $(f \mid j) \circ j = f$, as illustrated by

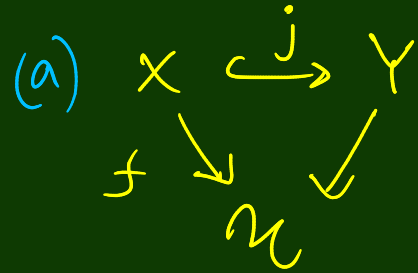
$$\begin{array}{ccc} X & \xhookrightarrow{j} & Y \\ f \searrow & & \swarrow f \mid j \\ & D & \end{array}$$

Some examples

MSCS'2021

(They need univalence.)

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



$$(f|j)(y) := \prod_{(x, -) : \text{fiber } j \ y} f(x) \cdot$$

(Right Kan extension.)

(b)

Use \sum instead.

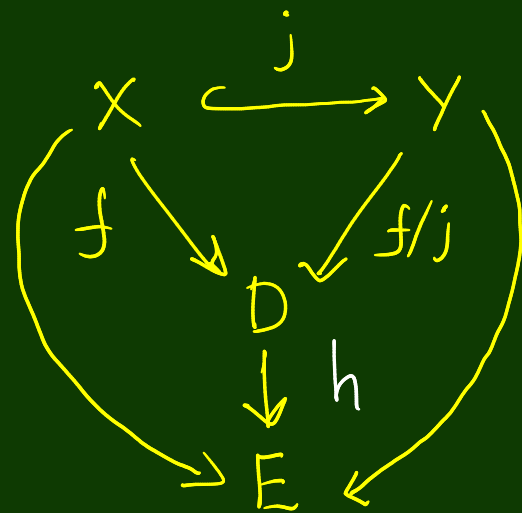
(Left Kan extension.)

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.

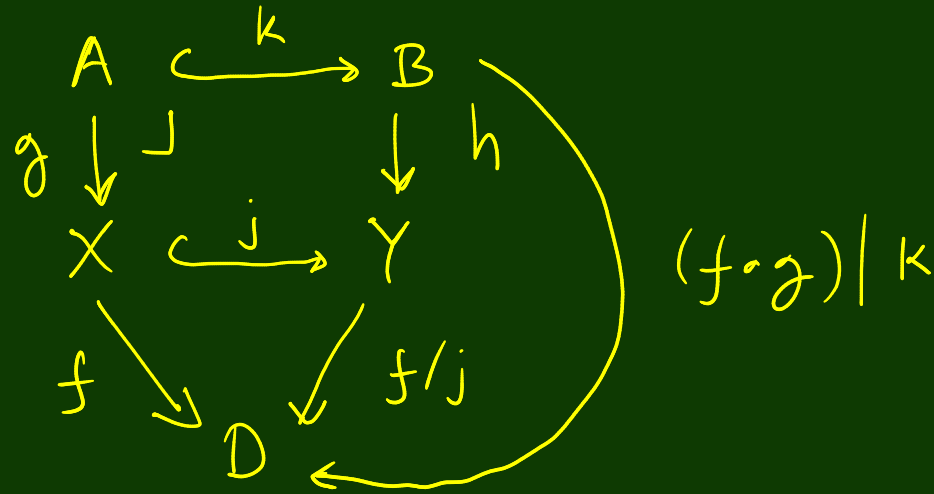
Homomorphisms of algebraic injectives



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

Pullback naturality

Previous examples are all pullback natural.



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

It is essential that the square is a pullback.

Consider the non-pullback square

$$\begin{array}{ccc} 0 & \xrightarrow{\quad} & 1 \\ \downarrow & & \downarrow \\ 1 & \xrightarrow{\quad} & 1 \end{array}$$

for a counter-example.

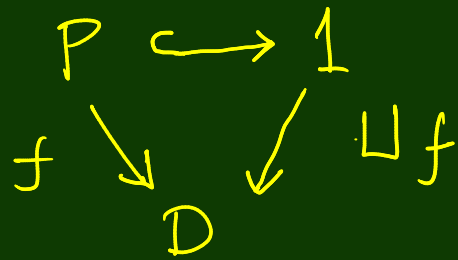
Associativity

$$\begin{array}{ccccc}
 X & \xhookrightarrow{j} & Y & \xhookrightarrow{k} & Z \\
 & \searrow f & \downarrow f|_j & \swarrow & \\
 & & D & &
 \end{array}
 \quad f|(k \circ j) = (f|_j)|_k$$

Examples (Mscs' 2024) $D := \mathcal{U}$ with extension given by Π or Σ .

II. Algebraic floppy structure

MSCS'2021



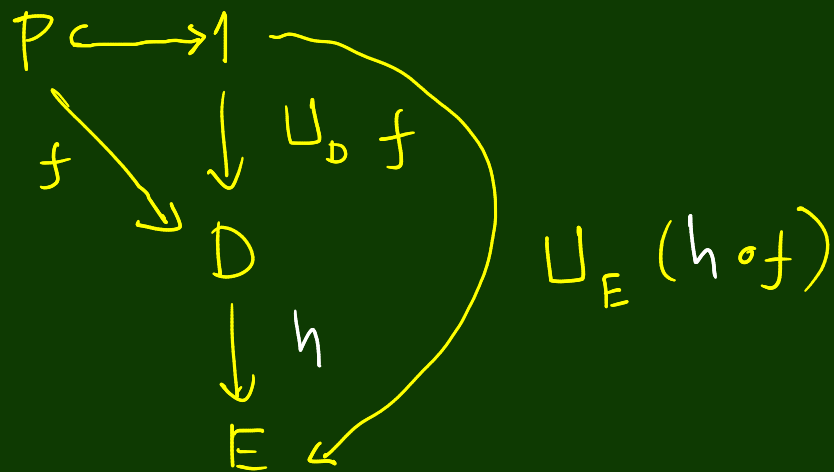
Every partial element
of D can be extended
to a total element.

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

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

Trivial fact. Algebraic inductive structure is, in particular,
an algebraic floppy structure.

Homomorphisms



$$h \circ U_D f = U_E(h \circ f)$$

construction | Algebraic fibby \rightarrow algebraic injective | MSCS'2021

$$\begin{array}{ccc}
 X & \xrightarrow{j} & Y \\
 f \searrow & & \swarrow \\
 & D &
 \end{array}
 \quad (f|j)(\gamma) := \bigsqcup_{(x,-) \in \underbrace{\text{fiber } j}_Y} f x$$

Fact (new observation)

This algebraic injective structure is pullback natural.
(Not only fiber-natural)

$$\begin{array}{ccc}
 P_Y & \longrightarrow & 1 \\
 p_{r_1} \downarrow & & \downarrow \bigsqcup (f \circ p_{r_1}) \\
 X & \xrightarrow{f} & D
 \end{array}$$

Fiberwise extension.

$$\begin{array}{ccc}
 P_Y & \xrightarrow{c} & 1 \\
 p_{r_1} \downarrow & \lrcorner & \downarrow y \\
 X & \xrightarrow{j} & Y \\
 f \searrow & & \swarrow f|j \\
 & D &
 \end{array}
 \quad \bigsqcup (f \circ p_{r_1})$$

Get this by fibbiness.

III. The lifting monad

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

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

The type of partial elements of X .

$$\begin{array}{ccc} X & & \\ \downarrow & & \\ X & \xrightarrow{f} & Y \\ \eta_X \downarrow & & \downarrow \eta_Y \\ \mathcal{L}X & \xrightarrow{\mathcal{L}f} & \mathcal{L}Y \end{array}$$

$$(\underline{1}, \lambda \cdot x) \quad (P, \varphi) \mapsto (P, f \circ \varphi)$$

$$\text{is-def(ined)} : \mathcal{L}X \rightarrow \Omega := p_{c1}$$

$$\text{value} : (\ell : \mathcal{L}X) \rightarrow \text{is-def } \ell \rightarrow X := p_{c2}$$

$$\begin{array}{ccc} X & \xrightarrow{g} & \mathcal{L}Y \\ \mathcal{L}X & \xrightarrow{g^\#} & \mathcal{L}Y \end{array}$$

$$(P, \varphi) \mapsto$$

is a prop,
as required.

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

Monad algebras

1. Structure map

$$\sqcup : \mathcal{Z}A \rightarrow A$$

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

2. Unit law

$$\sqcup (\lambda(-:1). a) = a \text{ for every } a:A.$$

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

3. Associativity law

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

$$P : \Omega$$

$$Q : P \rightarrow \Omega$$

$$f : \sum_{p:P} Q_p \rightarrow A$$

Previously not counted for
when discussing injectivity

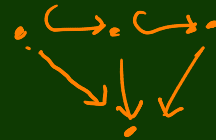
IV. Putting I-III together

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

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

Then \sqcup is associative iff \sqcap is associative

$$\sqcup \sqcup = \sqcup$$



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

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

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

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

But what about
extension data?
Ongoing.

Theorem. Let D be any type.

1. Then

$$\begin{aligned} & \text{pullback-natural, associative injective structure on } D \\ & \iff \text{associative algebraic flabby structure on } D \\ & = \text{ } \mathcal{L}\text{-algebra structure on } D. \end{aligned}$$

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

- What is missing to always have 3 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.

Try to define monad on types as follows.

1. Data

(a) A function $T : \text{Type} \rightarrow \text{Type}$

(May well change type levels.)

(b) A function $\eta : X \rightarrow TX$

(parametric in X .)

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

Then we can define

$$\mu : TTX \rightarrow TX := (\text{id}_{TX})^\#$$

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

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.

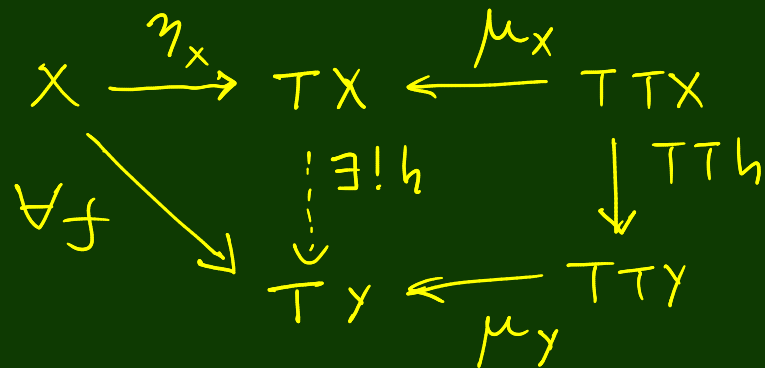
$$\begin{array}{ccccc}
 X & \xrightarrow{\eta_x} & TX & \xleftarrow{\mu_x} & TTX \\
 \searrow \forall f & & \downarrow \exists! h & & \downarrow TTh \\
 & & TY & \xleftarrow{\mu_y} & TTY
 \end{array}$$

The square says that h is an algebra homomorphism.

In HoTT/UF ,

$$(f: X \rightarrow TY) \rightarrow \text{is-contr} \left(\sum_{h: TX \rightarrow TY} \left(h \circ \eta_x \sim f \right) \times \left(h \circ \mu_x \sim \mu_y \circ T\eta \right) \right)$$

2.



In HoTT/UF ,

$$(f: X \rightarrow Ty) \rightarrow \text{is-contr} \left(\sum_{h: TX \rightarrow Ty} \left(h \circ \eta_x \sim f \right) \times \left(h \circ \mu_x \sim \mu_y \circ TTy \right) \right)$$

Likely problem.

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

The problem seems to disappear for idempotent monads

Tentative definition. An idempotent monad on types consists of

1. A function $T: \text{Type} \rightarrow \text{Type}$.

A family of functions $\eta_X: X \rightarrow TX$.

data
property

$$\begin{array}{ccc} X & \xrightarrow{\eta_X} & TX \\ \downarrow f & \searrow & \downarrow \exists! h \\ TY & & TY \end{array}$$

Here $\exists!$ is contractibility of Σ .

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

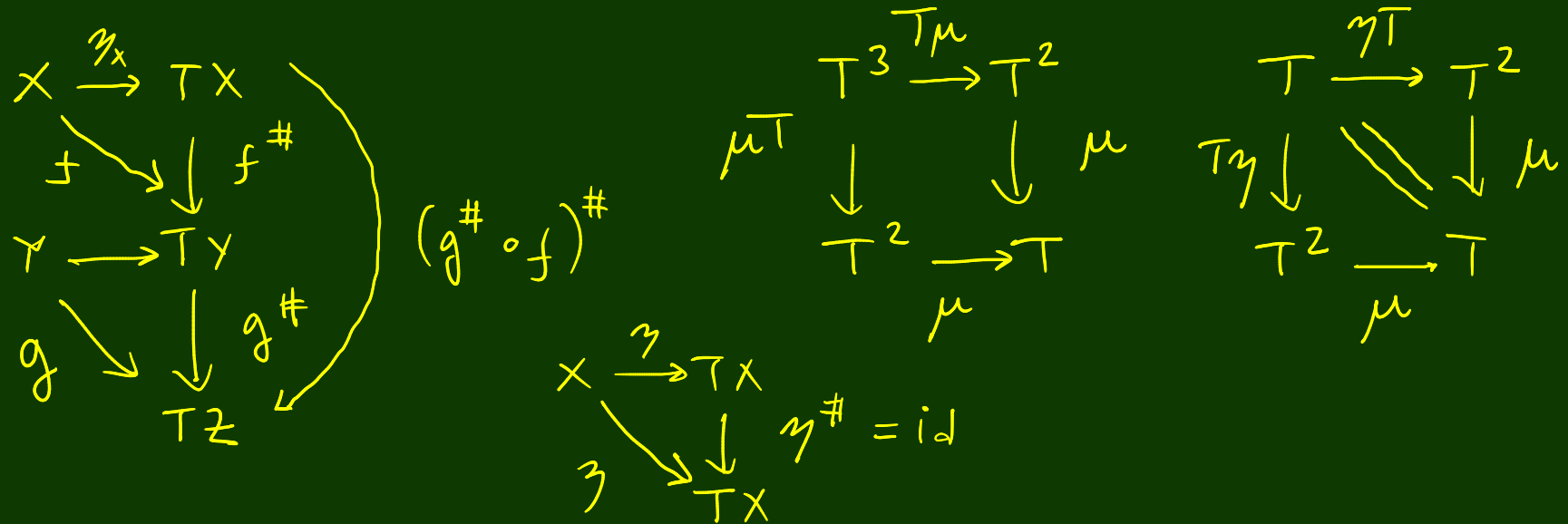
Idea: In an idempotent monad, every map of free algebras is a homomorphism

Then define $f^\# := \text{pr}_1 (\text{center of contraction } (\text{ext } f))$.
 $\nabla f := \text{pr}_2 (\text{center of contraction } (\text{ext } f))$.

That's the
complete
definition!

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

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



Idempotency

The maps

$$TX \xrightarrow[\eta_{TX}]{T\eta_X} TTX$$

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 \mathcal{V} and questions

1. This idea seems to work for **idempotent** monads on types.
2. We only give the data $T: \text{Type} \rightarrow \text{Type}$ and $\eta_X: X \rightarrow TX$.
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?