Semantics and syntax of higher inductive types

Michael Shulman[1] Peter LeFanu Lumsdaine[2]

[1]University of San Diego

[2]Stockholm University

http://www.sandiego.edu/~shulman/papers/stthits.pdf

March 20, 2016

Outline

My philosophy

- I want to use (“formal”) type theory as an internal language for higher categories.
- Therefore, I want a type theory that has semantics in a wide class of categories (not just one “intended” model).
- Today, we are semantically motivated: but the “intended semantics” is a large class of “good model categories”, which suffice (for instance) to represent every ∞-topos.
- In the distant future, it would be nice to be able to construct new models of HoTT inside HoTT. But for now, we use set-math (e.g. ZFC) as the metatheory.

The goal

Original Goal

Every good model category models higher inductive types.

The goal

Original Goal

Every good model category models higher inductive types.

Basic idea is 5 years old. Why not published yet?

1. We are easily distracted.
2. There are a lot of details in making it precise.
3. Easy to construct models of particular HITs; harder to say what a general "HIT" is.

Outline

Semantics of W-types

Definition

The W-type of $(x : A) \vdash B(x)$ type is inductively generated by

- $\mathsf{sup} : \prod_{(x:A)}(B(x) \to W_{A,B}) \to W_{A,B}$

or equivalently

- $\mathsf{sup} : \left(\sum_{(x:A)}(B(x) \to W_{A,B})\right) \to W_{A,B}$

Semantics of W-types

Definition

The W-type of $(x : A) \vdash B(x)$ type is inductively generated by

- $\mathrm{sup} : \prod_{(x:A)}(B(x) \to W_{A,B}) \to W_{A,B}$

or equivalently

- $\mathrm{sup} : \left(\sum_{(x:A)}(B(x) \to W_{A,B})\right) \to W_{A,B}$

Theorem (Classical)

$W_{A,B}$ is the initial algebra for the polynomial endofunctor

$$P_{A,B}(X) :\equiv \left(\sum_{(x:A)}(B(x) \to X)\right)$$

Now in category theory

Definition

The polynomial endofunctor associated to an exponentiable map $f : B \to A$ is

$$\mathcal{C} \xrightarrow{B^*} \mathcal{C}/B \xrightarrow{\Pi_f} \mathcal{C}/A \xrightarrow{\Sigma} \mathcal{C}$$

Definition

An algebra for an endofunctor $S : \mathcal{C} \to \mathcal{C}$ is an object X equipped with a map $SX \to X$.

How can we construct an initial algebra for an endofunctor?

Some categorical technology

- G. M. Kelly, "A unified treatment of transfinite constructions for free algebras, free monoids, colimits, associated sheaves, and so on", Bull. Austral. Math. Soc. 22 (1980), 1–83

Some categorical technology

- G. M. Kelly, "A unified treatment of transfinite constructions for free algebras, free monoids, colimits, associated sheaves, and so on", Bull. Austral. Math. Soc. 22 (1980), 1–83

Theorem (Kelly)

Let $\mathcal{A}$ be a cocomplete category with two cocomplete factorization systems $(\mathcal{E}, \mathcal{M})$ and $(\mathcal{E}', \mathcal{M}')$, let $\mathcal{A}$ be $\mathcal{E}$- and $\mathcal{E}'$-cowellpowered, let S be a well-pointed endofunctor, and for some regular cardinal α let S preserve the $\mathcal{E}'$-tightness of $(\mathcal{M}, \alpha)$-cones. Then S-Alg is constructively reflective in $\mathcal{A}$.

The high technology: user-friendly version

Theorem (Kelly?)

Let $\mathcal{C}$ be a locally presentable category. Then:

- *Every accessible endofunctor of $\mathcal{C}$ generates an algebraically-free monad.*
- *Every small diagram of accessible monads on $\mathcal{C}$ has an algebraic colimit.*

Review about monads

- Monad = endofunctor T with $\mu : TT \to T, \eta : \mathsf{Id} \to T$, axioms
- T-algebra = object X with $TX \to X$, axioms
- The forgetful functor $U_T : T\text{-Alg} \to \mathcal{C}$ has a left adjoint

$$F_T X = (TX, \mu_X : TTX \to TX).$$

and in particular an initial object $F_T(\emptyset)$.

Review about monads

- Monad = endofunctor T with $\mu : TT \to T, \eta : \mathsf{Id} \to T$, axioms
- T-algebra = object X with $TX \to X$, axioms
- The forgetful functor $U_T : T\text{-Alg} \to \mathcal{C}$ has a left adjoint

$$F_T X = (TX, \mu_X : TTX \to TX).$$

and in particular an initial object $F_T(\emptyset)$.

- The assignation $T \mapsto T\text{-Alg}$ is a fully faithful embedding

$$\mathsf{Monads}^{\mathsf{op}} \hookrightarrow \mathsf{Cat}_{/\mathcal{C}}.$$

i.e. we have

$$\mathsf{Monads}(T_1, T_2) \quad \cong \quad \mathsf{Cat}_{/\mathcal{C}}(T_2\text{-Alg}, T_1\text{-Alg})$$

Free monads

Definition

Every monad has an underlying endofunctor; this defines a functor

$$\text{monads on } \mathcal{C} \longrightarrow \text{endofunctors on } \mathcal{C}.$$

A free monad on an endofunctor S is the value at S of a (partially defined) left adjoint to this:

$$\mathrm{Monads}(\overline{S}, T) \cong \mathrm{Endofrs}(S, T)$$

$$\begin{tikzcd} S \arrow[r] \arrow[dr] & \overline{S} \arrow[d, dotted] \\ & T \end{tikzcd}$$

Algebraically-free monads

Definition

A monad $\overline{S}$ is algebraically-free on S if we have an equivalence of categories over $\mathcal{C}$:

$$\overline{S}\text{ monad-algebras} \xrightarrow{\simeq} S\text{ endofunctor-algebras}$$

Theorem (Kelly?)

Every algebraically-free monad is free, and the converse holds if $\mathcal{C}$ is locally small and complete.

Semantics of W-types, again

Idea

- Given $(x : A) \vdash B(x)$ type
- It interprets as a fibration $f : B \to A$, hence exponentiable
- The associated polynomial endofunctor S_f is accessible
- By Kelly's theorem, it generates an algebraically-free monad T_f
- Define $W_{A,B} = T_f(\emptyset)$.

Semantics of W-types, again

Idea

- Given $(x : A) \vdash B(x)$ type
- It interprets as a fibration $f : B \to A$, hence exponentiable
- The associated polynomial endofunctor S_f is accessible
- By Kelly's theorem, it generates an algebraically-free monad T_f
- Define $W_{A,B} = T_f(\emptyset)$.

Subtleties (ignore for now)

- Pullback-stability
- Fibrancy in homotopical models

Outline

Semantics of inductive types

Example

Consider the inductive type H generated by

- $\mathsf{sup}_1 : \prod_{(x:A)}(B(x) \to H) \to H$
- $\mathsf{sup}_2 : \prod_{(x:C)}(D(x) \to H) \to H$

Expect H to be initial among objects X equipped with two maps $\left(\sum_{(x:A)}(B(x) \to X)\right) \to X$ and $\left(\sum_{(x:C)}(D(x) \to X)\right) \to X$.

Questions

1. Given endofunctors S_1, S_2, is there a monad T whose algebras have (unrelated) S_1-algebra and S_2-algebra structures?
2. Given monads T_1, T_2, is there a monad T whose algebras have (unrelated) T_1-algebra and T_2-algebra structures?

Algebraic colimits of monads

Definition

An algebraic colimit of a diagram $D : J \to \mathrm{Monads}$ is a monad T with an equivalence of categories over $\mathcal{C}$:

$$T\text{-Alg} \xrightarrow{\simeq} \lim_{j \in J} \ D_j\text{-Alg}$$

This is a limit in $\mathrm{Cat}_{/\mathcal{C}}$, so it means that

T-algebra structures on X $\longleftrightarrow$ compatible families of D_j-algebra structures on X.

Theorem

Every algebraic colimit is a colimit in the category of monads, and the converse holds if $\mathcal{C}$ is locally small and complete.

Semantics of inductive types, again

Idea

- Each constructor yields a polynomial endofunctor, hence an algebraically-free monad T_c
- Take the algebraic coproduct $T = \sum_c T_c$ of all these monads
- The inductive type is $T(\emptyset)$.

Remark 1

The domains of constructors can be more general than in a W-type, but they can be reduced easily to that form using Σ-types.

Remark 2

The initial monad (the empty algebraic coproduct) is Id, whose algebras are just objects. Thus, the empty type — the inductive type generated by no constructors — is the initial object.

Outline

A first example

Example

Consider the propositional truncation $\|A\|$, generated by

- $A \to \|A\|$
- $\prod_{(x,y:\|A\|)}(x = y)$

- First constructor adds a point for every point of A
  $\rightsquigarrow$ constant endofunctor $S_1(X) = A$
- Second constructor adds a path for every two points of $\|A\|$
  $\rightsquigarrow$ endofunctor $S_2(X) = X \times X \times \mathtt{I}$
- How do we control the endpoints of those paths?

The boundary endofunctor

Define another endofunctor

$$\partial S_2(X) = X \times X \times \mathbf{2} = (X \times X) + (X \times X).$$

- A ∂S_2-algebra is a type with two binary operations.
- Every S_2-algebra is a ∂S_2-algebra via $\mathbf{2} \to \mathtt{I}$ (i.e. "take the endpoints").
- Every object is also a ∂S_2-algebra via $[\pi_1, \pi_2]$ (i.e. $(x, y) \mapsto x$ and $(x, y) \mapsto y$).
- The endpoints of the paths in an S_2-algebra are correct iff these two ∂S_2-algebra structures are the same.

Gluing intervals onto monads

So we are interested in the pullback category on the left:

$$\begin{tikzcd} \bullet \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{C} \arrow[d] \\ S_2\text{-Alg} \arrow[r] & \partial S_2\text{-Alg} \end{tikzcd} \qquad \begin{tikzcd} \overline{\partial S_2} \arrow[r] \arrow[d] & \mathrm{Id} \arrow[d] \\ \overline{S_2} \arrow[r] & T_2 \arrow[ul, phantom, "\ulcorner", very near start] \end{tikzcd}$$

which corresponds to the algebraic colimit of monads on the right.
We also need the S_1-algebra structure (a map from A), so:

Conclusion

$\|A\|$ is the initial $(\overline{S_1} + T_2)$-algebra.

Dependency between constructors

Example

The free group on A is generated by

- $A \to FA$
- $(\_ \cdot \_) : FA \to FA \to FA$
- $\prod_{(x,y,z:FA)} \left(x \cdot (y \cdot z) = (x \cdot y) \cdot z\right)$
- $\ldots$

NB: The source and target of the path in the third constructor (associativity) refer to the second constructor (multiplication).

- $S_1(X) = A$
- $S_2(X) = X \times X$
- $S_3(X) = X \times X \times X \times \mathtt{I}$, but then...

Dependency between constructors

$$S_3(X) = X \times X \times X \times \mathtt{I}$$
$$\partial S_3(X) = X \times X \times X \times \mathbf{2} = (X \times X \times X) + (X \times X \times X)$$

$$\begin{tikzcd}
\overline{\partial S_3} \arrow[r] \arrow[d] & \overline{S_2} \arrow[d] \\
\overline{S_3} \arrow[r] & \bullet \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

- Not every object is a ∂S_3-algebra in the right way...
- ...only the S_2-algebras are!
- The functor S_2-Alg $\to \partial S_3$-Alg equips an S_2-algebra (a "magma") with the two ternary operations "$x \cdot (y \cdot z)$" and "$(x \cdot y) \cdot z$".

Trivial dependency

For $\|A\|$, instead of a pushout and then a coproduct, we could instead consider the outer pushout:

$$\begin{tikzcd}
\overline{\partial S_2} \arrow[r] \arrow[d] \arrow[dr, phantom, "\ulcorner" very near end] & \mathrm{Id} \arrow[r] \arrow[d] \arrow[dr, phantom, "\ulcorner" very near end] & \overline{S_1} \arrow[d] \\
\overline{S_2} \arrow[r] & T_2 \arrow[r] & \overline{S_1} + T_2
\end{tikzcd}$$

Similarly, for the free group we could incorporate S_1 from the beginning:

$$\begin{tikzcd}
 & \mathrm{Id} \arrow[r] \arrow[d] \arrow[dr, phantom, "\ulcorner" very near end] & \overline{S_1} \arrow[d] \\
\overline{\partial S_3} \arrow[r] \arrow[d] \arrow[dr, phantom, "\ulcorner" very near end] & \overline{S_2} \arrow[r] \arrow[d] \arrow[dr, phantom, "\ulcorner" very near end] & \overline{S_1} + \overline{S_2} \arrow[d] \\
\overline{S_3} \arrow[r] & \bullet \arrow[r] & \bullet
\end{tikzcd}$$

Cofibrations

Question: What is special about $\mathbf{2} \to \mathtt{I}$ that makes this work?

$$\begin{tikzcd} S(X) \times \mathbf{2} \arrow[r, "u{,}v"] \arrow[d] & X \\ S(X) \times \mathtt{I} \arrow[ur, dotted] & \end{tikzcd} \iff \begin{tikzcd} & X^{\mathtt{I}} \arrow[d, two heads, "\text{fibration}"] \\ S(X) \arrow[r, "u{,}v"'] \arrow[ur, dotted] & X^{\mathbf{2}} \end{tikzcd}$$

$$\iff \prod_{s:S(X)} (u(s) =_X v(s))$$

Answer: $\mathbf{2} \to \mathtt{I}$ is a cofibration.

Other cofibrations

- If $C \to D$ is a cofib. & X is fibrant, $X^D \to X^C$ is a fibration.
- Hence we have types (its fibers) of "(strict) extensions of a given map $C \to X$ to a map $D \to X$."
- Other cofibrations give other kinds of constructors:

$$\begin{array}{ccc}
\emptyset \to \mathbf{1} & \rightsquigarrow & X \\
\mathbf{2} \to \mathtt{I} & \rightsquigarrow & x =_A y \\
\partial G_2 \to G_2 & \rightsquigarrow & p =_{x=y} q \\
\partial \square^2 \to \square^2 & \rightsquigarrow & \mathrm{Square}(p,q,r,s) \\
\vdots & & \vdots
\end{array}$$

Outline

From semantics to syntax

Definition

A HIT spec consists of

- An ordered list of constructors.
- Each constructor has a domain, giving a polynomial endofunctor S_n.
- Each constructor has a shape, which is a cofibration $C_n \to D_n$ mapping the "boundary" into the "path".
- Finally, each constructor has a boundary, which has something to do with C_n.

The semantics of a HIT spec

- Each constructor yields a map of free monads

$$\overline{S_n \times C_n} \to \overline{S_n \times D_n}$$

- Starting from $T_0 = \mathrm{Id}$, we build up monads successively:

$$\begin{tikzcd} \overline{S_n \times C_n} \arrow[r] \arrow[d] & T_{n-1} \arrow[d] \\ \overline{S_n \times D_n} \arrow[r] & T_n \arrow[ul, phantom, "\ulcorner", very near start] \end{tikzcd}$$

A monad built in this way we call a cell monad.

- A HIT with n constructors is $T_n(\emptyset)$.

Those pesky boundaries

Question

What can the boundaries of a path-constructor be?

Answer

The semantics tells us! They have to be:

- monad morphisms $\overline{S_n \times C_n} \to T_{n-1}$, or equivalently
- endofunctor maps $S_n \times C_n \to T_{n-1}$.

But what are those?

Free endofunctors

Suppose S_n is polynomial on $(a : A_n) \vdash B_n(a)$:

$$\begin{aligned} S_n(X) &= \textstyle\sum_{a:A_n} X^{B_n(a)} \\ (S_n \times C_n)(X) &= \left(\textstyle\sum_{a:A_n} X^{B_n(a)}\right) \times C_n \\ &= \textstyle\sum_{(a,c):A_n \times C_n} X^{B_n(a)} \end{aligned}$$

- Internally, this is a coproduct of the functors $\lambda X.X^{B_n(a)}$. So $S_n \times C_n \to T_{n-1}$ consists of "a map $\lambda X.X^{B_n(a)} \to T_{n-1}$ for each $(a, c) : A_n \times C_n$".

Free endofunctors

Suppose S_n is polynomial on $(a : A_n) \vdash B_n(a)$:

$$\begin{aligned} S_n(X) &= \textstyle\sum_{a:A_n} X^{B_n(a)} \\ (S_n \times C_n)(X) &= \left(\textstyle\sum_{a:A_n} X^{B_n(a)}\right) \times C_n \\ &= \textstyle\sum_{(a,c):A_n \times C_n} X^{B_n(a)} \end{aligned}$$

- Internally, this is a coproduct of the functors $\lambda X.X^{B_n(a)}$. So $S_n \times C_n \to T_{n-1}$ consists of "a map $\lambda X.X^{B_n(a)} \to T_{n-1}$ for each $(a, c) : A_n \times C_n$".
- But by (internal) Yoneda,

$$\mathrm{NatTrans}(\lambda X.X^{B_n(a)}, T_{n-1}) = T_{n-1}(B_n(a)).$$

- So natural transformations $S_n \times C_n \to T_{n-1}$ are the same as

$$\textstyle\prod_{(a,c):A_n \times C_n} T_{n-1}(B_n(a))$$

Syntax for boundaries

$$\prod_{(a,c):A_n \times C_n} T_{n-1}(B_n(a))$$

Syntax for boundaries

$$\prod_{a:A_n} \left(C_n \to T_{n-1}(B_n(a)) \right)$$

Syntax for boundaries

$$\prod_{a:A_n} \Big(C_n \to T_{n-1}(B_n(a)) \Big)$$

- $T_{n-1}(\emptyset)$ is the HIT generated by the first $(n-1)$ constructors.
- $T_{n-1}(B_n(a))$ is the HIT generated by the first $(n-1)$ constructors and one extra constructor with domain $B_n(a)$.

Thus, the boundary of a constructor $\prod_{a:A_n} \prod_{f:B_n(a)\to W} \cdots$ is

- For each $a : A_n$,
- . . . a C_n-shaped picture (pair of points, parallel paths, etc.)
- . . . in the HIT generated by the previous constructors and new symbols "$f(b)$" for all $b : B_n(a)$.

Examples

loop : base = base	"base" is a term in the HIT generated by "base" only
merid : $\prod_x N = S$	N and S are terms in the HIT generated by N and S only
surf : $p \cdot q = q \cdot p$	$p \cdot q$ and $q \cdot p$ are terms in the HIT generated by b, $p : b = b$, and $q : b = b$
$\prod_{x,y:\|A\|}(x = y)$	x and y are interpreted as "$f(b)$": each is a term in the HIT generated by A and $\mathbf{1}$ (here $A = \mathbf{2}$, $B(a) = \mathbf{1}$)

A more complicated example

In the localization $L_f(A)$ at $f : P \to Q$, we see:

$$\prod_{x:P} \prod_{g:P \to L_f(A)} \mathrm{ext}(g, f(x)) = g(x)$$

Here $A = P$, $B(a) = P$, and both $\mathrm{ext}(g, f(x))$ and $g(x)$ are (assuming $x : P$) terms in the HIT W' generated by $\mathrm{ext} : \prod_{g:P \to W'}(Q \to L_f(A))$ and $g : P \to W'$.

Stepping back

- In general, we expect some "grammar" describing what the boundary of a constructor can be.
- We are leveraging the type theory itself to be this grammar: the boundary simply consists of terms in a particular type.

HIT specs, again

Definition

Inductively, a HIT spec W is either empty, or consists of:

- A HIT spec W' (the previous constructors).
- A constructor domain $(a : A) \vdash B(a)$.
- A constructor shape, which is a cofibration $C \to D$.
- A constructor boundary $\prod_{a:A} \left(C \to W'_a\right)$, where W'_a is the HIT generated by W' together with a map $B(a) \to W'_a$.

Rules for HITs

Definition

Given a HIT spec W, a W-algebra is a type X together with:

- A W'-algebra structure (inductively), and...
- For each $a : A$ and $f : B(a) \to X$, the W'-algebra structure and f make X a W'_a-algebra. So by recursion we have $W'_a \to X$, hence a boundary composite $C \to W'_a \to X$. The additional data is an extension of this to D:

$$\begin{tikzcd} C \arrow[r] \arrow[d] & W'_a \arrow[r] & X \\ D \arrow[urr, dotted] & & \end{tikzcd}$$

Rules for HITs

Definition

Given a HIT spec W, a W-algebra is a type X together with:

- A W'-algebra structure (inductively), and...
- For each $a : A$ and $f : B(a) \to X$, the W'-algebra structure and f make X a W'_a-algebra. So by recursion we have $W'_a \to X$, hence a boundary composite $C \to W'_a \to X$. The additional data is an extension of this to D.

- Intro: "W is a W-algebra".
- Elim: "Any dependent W-algebra over W has a section."
- Comp: "The section is a W-algebra map."

Outline

What are these cofibrations, anyway?

We could either

1. Fix a particular set of cofibrations that exist in models of interest, like $\partial G_n \to G_n$ or $\partial\square^n \to \square^n$.
2. Extend the type theory with a judgment for cofibrations, and a "type of extensions" of a given function along such a cofibration.
    - In the case $\mathbf{2} \to \mathtt{I}$ this will behave like cubical identity tyes.
    - NB: objects like $\mathtt{I}$ are not usually fibrant; put them in a separate context (like cubical "dimension variables") or use HTS-style "pretypes".

The type of extensions along a cofibration

$$\frac{\Gamma \vdash i : A \rightarrowtail B \qquad \Gamma, y : B \vdash C \text{ type} \qquad \Gamma, x : A \vdash d : C[i(x)/y]}{\Gamma \vdash \mathsf{Extn}_{i,y.C}(x.d) \text{ type}}$$

$$\frac{\Gamma, y : B \vdash c : C \qquad \Gamma, x : A \vdash c[i(x)/y] \equiv d}{\Gamma \vdash \mathring{\lambda} y.\, c : \mathsf{Extn}_{i,y.C}(x.d)}$$

$$\frac{\Gamma \vdash f : \mathsf{Extn}_{i,y.C(y)}(x.d(x)) \qquad \Gamma \vdash b : B}{\Gamma \vdash f @ b : C(b) \qquad f @ (i(a)) \equiv d[a/x]}$$

(plus β, η)

- For $1 + 1 \rightarrowtail \mathtt{I}$, reproduces cubical identity types.
- Semantically, represents the pullback corner map (Leibniz cotensor) of a cofibration against a fibration.

What's up with that induction?

Question

HITs are defined inductively. Where does that induction happen?

Answer #1

In the metatheory.

I.e. given any type theory containing some HITs, we can choose one of them, choose a domain, shape, and boundary to determine a new constructor, and obtain a new type theory containing one more HIT.

A theory containing HITs

Answer #2

By defining a new judgment form inside the theory.

I.e. we have a judgment for “HIT specs”, whose rule is “add a new constructor”, and a rule that any HIT spec gives a HIT.

But now

Any judgment form in the theory must be interpreted by something in the semantics. What is a “semantic HIT spec”?

A theory containing HITs

Answer #2

By defining a new judgment form inside the theory.

I.e. we have a judgment for "HIT specs", whose rule is "add a new constructor", and a rule that any HIT spec gives a HIT.

But now

Any judgment form in the theory must be interpreted by something in the semantics. What is a "semantic HIT spec"?

. . . a monad.

A type theory with monads

Judgment	Meaning
$\Gamma \vdash A$ type	A is a type in type context Γ
$\Gamma \vdash a : A$	a is a term of type A in type context Γ
$\Gamma \vdash T$ monad	T is a monad in type context Γ
$\Gamma \mid \tau : T \vdash s : S$	$\tau.s$ notates a monad morphism $T \to S$
$\Gamma \mid \tau : T \parallel \Delta \vdash A$ type	A is a T-algebra
$\Gamma \mid \tau : T \parallel \Delta \vdash a : A$	a is a T-algebra morphism $\Delta \to A$

A type theory with monads

Judgment	Meaning
$\Gamma \vdash A$ type	A is a type in type context Γ
$\Gamma \vdash a : A$	a is a term of type A in type context Γ
$\Gamma \vdash T$ monad	T is a monad in type context Γ
$\Gamma \mid \tau : T \vdash s : S$	$\tau.s$ notates a monad morphism $T \to S$
$\Gamma \mid \tau : T \parallel \Delta \vdash A$ type	A is a T-algebra
$\Gamma \mid \tau : T \parallel \Delta \vdash a : A$	a is a T-algebra morphism $\Delta \to A$

- Substitution for type variables in monads
  $\Rightarrow$ all monads are indexed
- Substitution for monad variables in algebras
  $\Rightarrow$ a monad map $T \to S$ gives a functor S-Alg $\to$ T-Alg
- Elimination of monad formers into all judgments
  $\Rightarrow$ monad colimits are algebraic colimits

Part II

All the lies I just told you

Outline

Problem #1: Fibrancy

Problem

$T(\emptyset)$ may not be fibrant, hence may not represent a type.

- In non-recursive cases (e.g. empty type, coproduct type) we can just fibrantly replace it.
- But in recursive cases, its fibrant replacement may no longer be a T-algebra: the newly added fillers need a free T-action, which may produce new horns that need fillers, etc.

Building in fibrancy

Solution

1. Let R be the fibrant replacement monad (Garner).
2. Let $T_R = T + R$.

- A T_R-algebra is a T-algebra with an unrelated R-algebra structure.
- In particular, every T_R-algebra is fibrant!

Theorem (L-S)

$T_R(\emptyset)$ satisfies the T-algebra induction principle: any fibration $p : Y \to T_R(\emptyset)$ that is a T-algebra map has a T-algebra section.

Outline

Problem #2: Pullback-stability

Problem

Everything in a model of type theory must be strictly stable under pullback.

- Use indexed endofunctors, monads, free monads, colimits of monads. Everything is pullback-stable up to iso...
- ...except R, which is not an indexed monad at all!
- Local universes mumbo-jumbo: form $T + R$ in the "universal case."

Outline

Problem #3: Boundaries can't use fibrancy

Problem

R is not an indexed monad, but our algebraic colimits have to work on indexed monads.

- Therefore, we have to coproduct with R at the very end to obtain our HIT.
- Therefore, in the middle, the "previous constructors" HIT is not fibrant.
- Therefore, we can't use "fibrant operations" (like path concatenation and eliminators) in boundaries of constructors.

Problem #3: Boundaries can't use fibrancy

Problem

R is not an indexed monad, but our algebraic colimits have to work on indexed monads.

- Therefore, we have to coproduct with R at the very end to obtain our HIT.
- Therefore, in the middle, the "previous constructors" HIT is not fibrant.
- Therefore, we can't use "fibrant operations" (like path concatenation and eliminators) in boundaries of constructors.

Partial workaround(s)

- Hub and spoke does not help.
- Use cofibrations like $\partial\square \to \square$ whose domains implicitly involve "concatenations"

Outline

Problem #4: Closure of universes

Problem

Even in the best model categories, we don't know how to show that any universes are closed under HITs.

- A universe $\widetilde{U} \to U$ classifies fibrations with "small fibers".
- For closure of U, the "universal HIT" over something built from U must be classified by U.
- But the (algebraic) fibrant replacement of a map with small fibers over a large base may no longer have small fibers!