Model Structures on Types in Type Theory

(joint work with Nicolas Tabareau)

Simon boulier

Model Structures on Types in Type Theory

(joint work with Nicolas Tabareau)

Unfortunately Not Here

Motivations

1. Understand better the connection between Type, Top, sSet, ∞Gpd

2. challenge / benchmark for a 2-level type theory (e.g., HTS)

3. Use this model structure to define homotopy limits and colimits

Part I. Model Structures in a type theory with a strict equality

Typing rules for Martin-Löf type theory with a strict equality

Usual rule of identity type

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \mathbf{refl}_t : t \equiv_A t} \qquad \frac{\begin{array}{c}\Gamma \vdash t, t' : A \qquad \Gamma \vdash e : t \equiv_A t' \\ \Gamma, y : A, q : t \equiv_A y \vdash P : \mathcal{U}_i \qquad \Gamma \vdash u : P\{y := t, q := \mathbf{refl}_t\}\end{array}}{\Gamma \vdash J_{\equiv}(A, y.q.P, t, t', e, u) : P\{y := t', q := e\}}$$

Specific rules for strictness

$$\frac{\Gamma \vdash f, g : \Pi\, x : A.\, B\, x \qquad \Gamma \vdash e : \Pi\, x : A.\, f\, x \equiv_{B\, x} g\, x}{\Gamma \vdash \mathbf{funext}_{\equiv}(e) : f \equiv_{\Pi\, x:A.\ B\, x} g} \qquad \frac{\Gamma \vdash e_1, e_2 : t \equiv_A t'}{\Gamma \vdash \mathbf{UIP}(e_1, e_2) : e_1 \equiv_{t \equiv t'} e_2}$$

Categories in MLTT with strict equality

- Definition A *category* consists of:
  - a type A of objects,
  - for all $a, b : A$, a type $\mathrm{Hom}(a, b)$ of arrows
  - for all $a : A$, an identity arrow $\mathbf{id}_a : \mathrm{Hom}(a, a)$
  - for all $a, b, c : A$, a composition function $\_ \circ \_ : \mathrm{Hom}(b, c) \to \mathrm{Hom}(a, b) \to \mathrm{Hom}(a, c)$

Categories in MLTT with strict equality

- Definition A *category* consists of:
  - a type A of objects,
  - for all $a, b : A$, a type $\mathrm{Hom}(a, b)$ of arrows
  - for all $a : A$, an identity arrow $\mathbf{id}_a : \mathrm{Hom}(a, a)$
  - for all $a, b, c : A$, a composition function $\_ \circ \_ : \mathrm{Hom}(b, c) \to \mathrm{Hom}(a, b) \to \mathrm{Hom}(a, c)$

Plus commutation "on the noze"

  - for all $f : \mathrm{Hom}(a, b)$, a proof of $f \circ \mathbf{id}_a \equiv f$ and $\mathbf{id}_b \circ f \equiv f$
  - for all $f : \mathrm{Hom}(a, b), g : \mathrm{Hom}(b, c), h : \mathrm{Hom}(c, d)$, a proof of $h \circ (g \circ f) \equiv (h \circ g) \circ f$.

Categories in MLTT with strict equality

- Definition A *category* consists of:
  - a type A of objects,
  - for all $a, b : A$, a type $\mathrm{Hom}(a, b)$ of arrows
  - for all $a : A$, an identity arrow $\mathbf{id}_a : \mathrm{Hom}(a, a)$
  - for all $a, b, c : A$, a composition function $\_ \circ \_ : \mathrm{Hom}(b, c) \to \mathrm{Hom}(a, b) \to \mathrm{Hom}(a, c)$

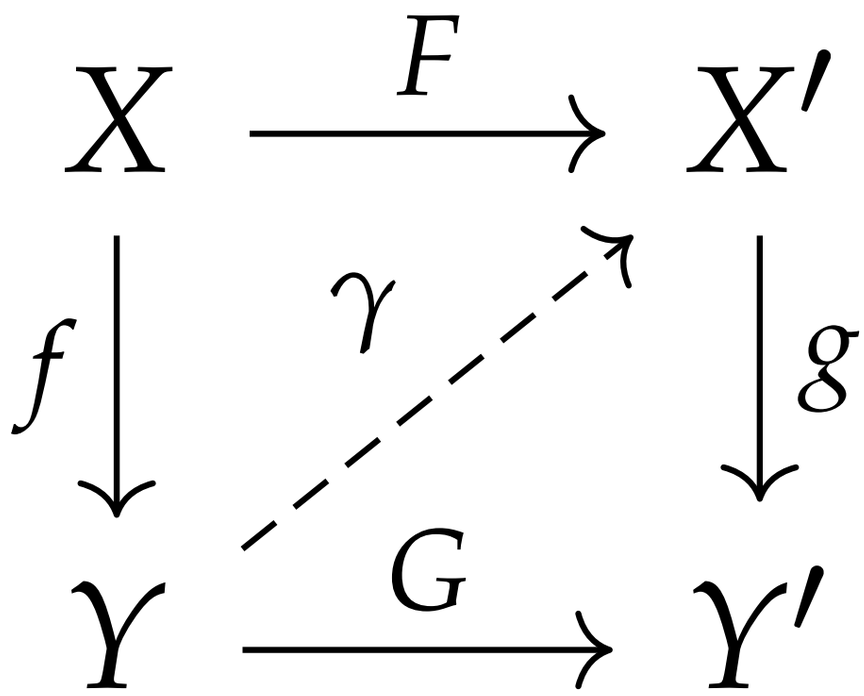
Plus commutation "on the noze"

- for all $f : \mathrm{Hom}(a, b)$, a proof of $f \circ \mathbf{id}_a \equiv f$ and $\mathbf{id}_b \circ f \equiv f$
- for all $f : \mathrm{Hom}(a, b), g : \mathrm{Hom}(b, c), h : \mathrm{Hom}(c, d)$, a proof of $h \circ (g \circ f) \equiv (h \circ g) \circ f$.

Type is a category

Type (and functions) forms a category because the laws hold definitionally (thanks to β-reduction).

Left/Right Lifting Properties



- f has LLP with respect to g
- g has RLP with respect to f

Weak factorisation system

▸ Definition A *weak factorization system* (wfs) on $\mathcal{C}$ consists of two classes of arrows L and R such that:

1. every arrow f of $\mathcal{C}$ can be factorized as $f \equiv r \circ l$ with $l \in L$ and $r \in R$
2. L is exactly the class of arrows of $\mathcal{C}$ which have the LLP with respect to R : $L \sim \mathbf{LLP}(R)$
3. R is exactly the class of arrows of $\mathcal{C}$ which have the RLP with respect to L : $R \sim \mathbf{RLP}(L)$

Model Structure

▸ Definition A *model structure* on $\mathcal{C}$ consists of three classes of arrows F, C and W (the *fibrations*, the *cofibrations* and the *weak equivalences*) such that:

1. W satisfies the 2-out-of-3 property
2. (AC, F) and (C, AF) are two weak factorization systems,

where $AC := C \cap W$ and $AF := F \cap W$.

- C : cofibrations
- F : fibrations
- AC : acyclic cofibrations
- AF : acyclic fibrations

Part II. Homotopy Type System

Homotopy Type System

1. HTS has first been introduced by VV.

2. It has been rephrased recently as a
   2-level type theory by Altenkirch et al.

[Voe13] Vladimir Voevodsky. A simple type system with two identity types, 2013.

[Alt16] Thorsten Altenkirch, Paolo Capriotti, Nicolai Kraus,
Extending Homotopy Type Theory with Strict Equality, CSL'16.

Homotopy Type System

To allow UIP and univalence in the same theory,
we need to guarantee:

$$a \equiv b \to a = b \quad \text{but} \quad a = b \nrightarrow a \equiv b.$$

Homotopy Type System

To allow UIP and univalence in the same theory,
we need to guarantee:

$$a \equiv b \to a = b \quad \text{but} \quad a = b \nrightarrow a \equiv b.$$

⇒ Introduce a notion of fibrant types

Identity path for fibrant types

$$\frac{\Gamma \vdash A : \mathcal{U}^{\mathbf{F}}\, i \qquad \Gamma \vdash t,\, t' : A}{\Gamma \vdash t =_A t' : \mathcal{U}_i}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \mathbf{1}_t : t =_A t}$$

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash A\ \mathbf{Fib}}{\Gamma \vdash A : \mathcal{U}_i^F}$$

$$\frac{\begin{array}{c}\Gamma \vdash t,\, t' : A \qquad \Gamma \vdash p : t =_A t' \\ \Gamma,\, y : A,\, q : t =_A y \vdash P\ \mathbf{Fib} \qquad \Gamma \vdash u : P\{y := t, q := \mathbf{1}_t\}\end{array}}{\Gamma \vdash J_=(A,\, y.q.P,\, t,\, t',\, p,\, u) : P\{y := t', q := p\}}$$

$$\frac{\Gamma \vdash A : \mathcal{U}_i^F}{\Gamma \vdash A : \mathcal{U}_i}$$

Identity path for fibrant types

$$\frac{\Gamma \vdash A : \mathcal{U}^{\mathbf{F}}\, i \qquad \Gamma \vdash t,\, t' : A}{\Gamma \vdash t =_A t' : \mathcal{U}_i}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \mathbf{1}_t : t =_A t}$$

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash A\ \mathbf{Fib}}{\Gamma \vdash A : \mathcal{U}_i^F}$$

$$\frac{\Gamma \vdash t,\, t' : A \qquad \Gamma \vdash p : t =_A t' \qquad \Gamma,\, y : A,\, q : t =_A y \vdash P\ \mathbf{Fib} \qquad \Gamma \vdash u : P\{y := t, q := \mathbf{1}_t\}}{\Gamma \vdash J_=(A,\, y.q.P,\, t,\, t',\, p,\, u) : P\{y := t', q := p\}}$$

$$\frac{\Gamma \vdash A : \mathcal{U}_i^F}{\Gamma \vdash A : \mathcal{U}_i}$$

Rules for fibrancy

$$\frac{}{\Gamma \vdash \mathcal{U}_i \ \mathbf{Fib}} \qquad \frac{}{\Gamma \vdash \mathcal{U}_i^F \ \mathbf{Fib}} \qquad \frac{\Gamma \vdash A : \mathcal{U}_i^F}{\Gamma \vdash A \ \mathbf{Fib}} \qquad \frac{\Gamma \vdash A \ \mathbf{Fib} \qquad \Gamma, x : A \vdash B \ \mathbf{Fib}}{\Gamma \vdash \Pi\, x : A.\, B \ \mathbf{Fib}}$$

$$\frac{\Gamma \vdash A \ \mathbf{Fib} \qquad \Gamma, x : A \vdash B \ \mathbf{Fib}}{\Gamma \vdash \Sigma\, x : A.\, B \ \mathbf{Fib}} \qquad \frac{\Gamma \vdash A \ \mathbf{Fib} \qquad \Gamma \vdash t, t' : A}{\Gamma \vdash t =_A t' \ \mathbf{Fib}}$$

Hack in Coq

We found a way to emulate HTS in Coq using type classes.
First we define a type class Fibrant to keep track of fibrant types:

```
Axiom dummy_fibrant_type : Type.
Class Fibrant (A: Type) := { dummy_fibrant_value : dummy_fibrant_type }.
```

And we postulate fibrancy rules. For instance:

```
Axiom fibrant_forall: ∀ (A:Type) (B: A → Type),
 Fibrant A → (∀ x, Fibrant (B x)) → Fibrant (∀ x, B x).
```

Hack in Coq

The identity type is defined as a private inductive type to forbid the use of its elimination principle when the predicate is not fibrant:

```
Private Inductive paths {A : Type} (x : A) : A → Type := idpath : paths x x.

Definition paths_ind (A : Type) (x : A) (P : ∀ y : A, paths x y → Type)
    (FibP : ∀ y p, Fibrant (P y p)) (u : P x idpath) (y : A) (p : paths x y) : P y p
:= match p with | idpath ⇒ u end.
```

The fibrancy conditions are checked ***automatically*** by type class inference.
The universe of fibrant types is defined using a coercion:

```
Record FType := { FType_T : Type;
                  FType_F : Fibrant FType_T}.

Coercion FType_T : FType ⟼ Sortclass.
```

Hack in Coq

The identity type is defined as a private inductive type to forbid the use of its elimination principle when the predicate is not fibrant:

```
Private Inductive paths {A : Type} (x : A) : A → Type := idpath : paths x x.

Definition paths_ind (A : Type) (x : A) (P : ∀ y : A, paths x y → Type)
    (FibP : ∀ y p, Fibrant (P y p)) (u : P x idpath) (y : A) (p : paths x y) : P y p
:= match p with | idpath ⇒ u end.
```

The fibrancy conditions are checked ***automatically*** by type class inference.
The universe of fibrant types is defined using a coercion:

```
Record FType := { FType_T : Type;
                  FType_F : Fibrant FType_T}.

Coercion FType_T : FType ⟼ Sortclass.
```

Part III. Model Structure on fibrant types

(AC,F)-WFS

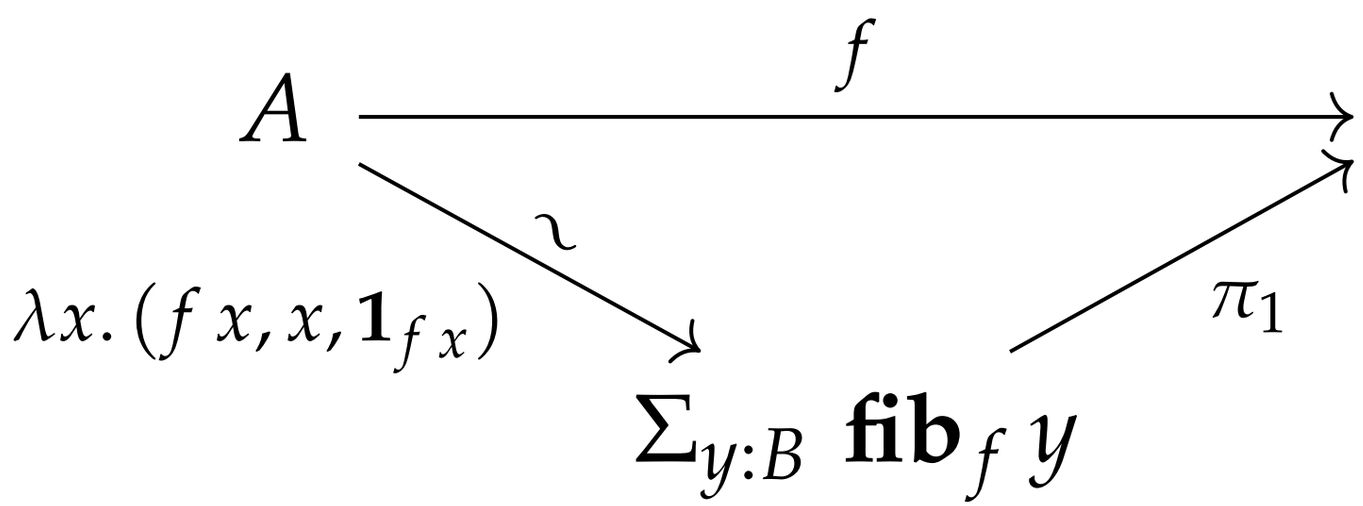
The definition of fibrations is based on fibrancy.

The factorisation system comes from the well known factorisation with the homotopy fiber as done in [Gam08].

[Gam08] Nicola Gambino and Richard Garner. The identity type weak factorisation system. *Theor. Comput. Sci.*, 409(1):94–109, 2008.

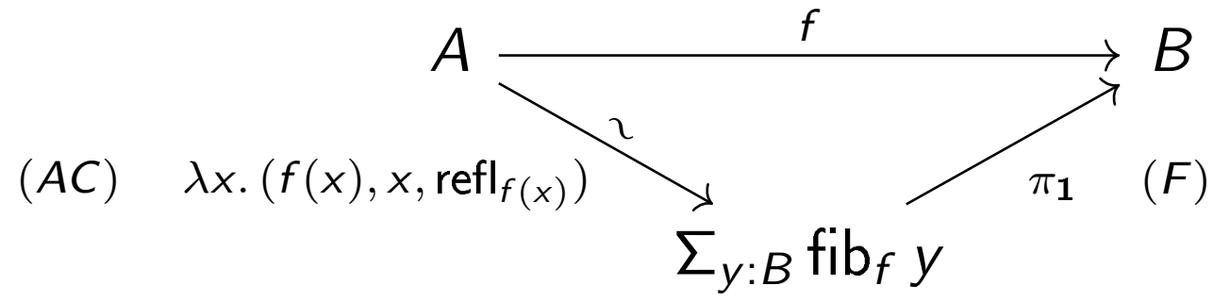
(AC,F)-WFS

▸ Definition A function $f : A \to B$ is said to be a *fibration* if there exists a **fibrant** type family $P : A' \to \mathcal{U}_i$ such that f is a retract of $\pi_1 : \Sigma_{x:A'} P\, x \to A'$.

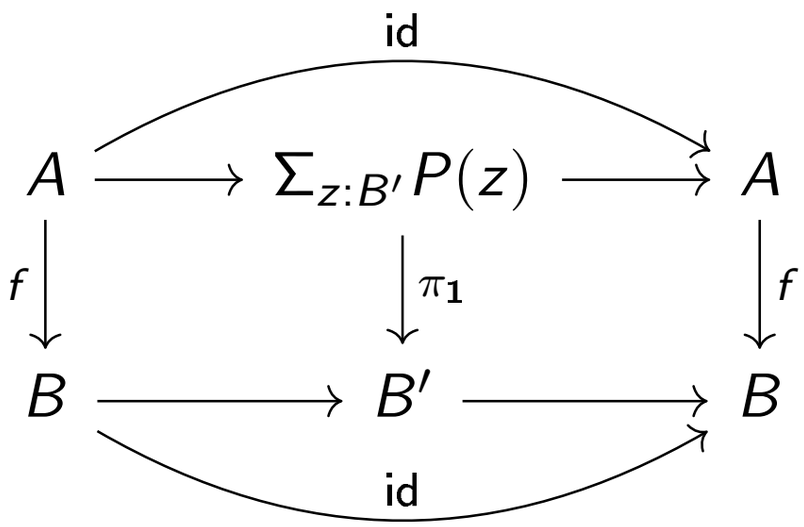


$$\mathbf{fib}_f := \lambda\, y.\, \Sigma\, x : A.\, f\, x = y$$

(AC,F)-WFS

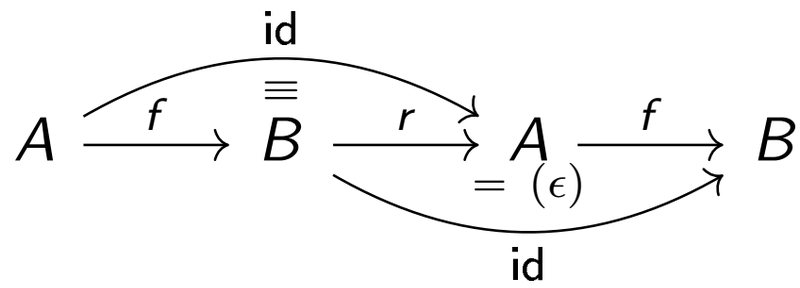


Fibrations:



with $P : B' \to \mathcal{U}_{\mathsf{F}}$

Injective Equivalences (AC):



$+\ \epsilon_{f(x)} \equiv \mathsf{refl}_{f(x)}$

(C,AF)-WFS

The definition of cofibrations is more tricky.

Its requires the introduction of an HIT: the cylinder as done in [Lum11]

[Lum11] Peter LeFanu Lumsdaine. *Model Structures from Higher Inductive Types*. Unpublished notes, 2011.

Cylinders

$$\frac{\Gamma \vdash t : B}{\Gamma \vdash \mathbf{Cyl}_f\, t : \mathcal{U}_i} \qquad \frac{\Gamma \vdash t : B}{\Gamma \vdash \mathbf{Cyl}_f\, t\ \mathbf{Fib}} \qquad \frac{}{\Gamma \vdash \mathbf{top}_f : \Pi\, x : A.\, \mathbf{Cyl}_f (f\, x)}$$

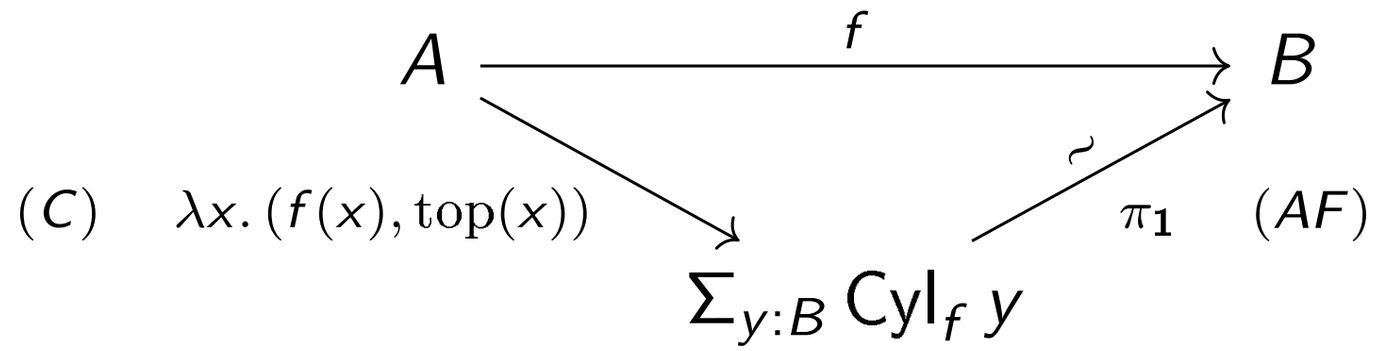
$$\frac{}{\Gamma \vdash \mathbf{base}_f : \Pi\, y : B.\, \mathbf{Cyl}_f\, y} \qquad \frac{}{\Gamma \vdash \mathbf{cyl\_eq}_f : \Pi\, x : A.\, \mathbf{top}_f\, x = \mathbf{base}_f\, (f\, x)}$$

Cylinders

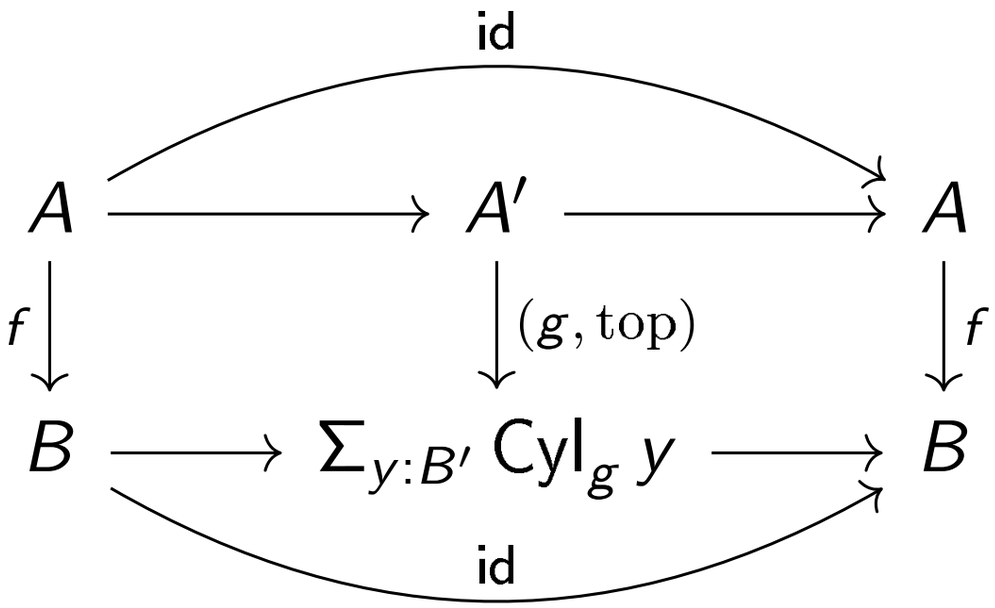
The elimination rule is given by:

$$\frac{\begin{array}{c}\Gamma,\, y : B,\, w : \mathbf{Cyl}_f\, y \vdash P\ \mathbf{Fib} \\ \Gamma \vdash \mathrm{top}' : \Pi\, x : A.\, P\,(f\,x)\,(\mathbf{top}\,x) \qquad \Gamma \vdash \mathrm{base}' : \Pi\, y : B.\, P\, y\,(\mathbf{base}\, y) \\ \Gamma \vdash \mathrm{cyl\_eq}' : \Pi\, x : A.\, (\mathbf{cyl\_eq}\, x)\ \#\ (\mathrm{base}'\,(f\,x)) = \mathrm{top}'\, x\end{array}}{\Gamma \vdash \mathbf{cyl\_ind}(P, \mathrm{top}', \mathrm{base}', \mathrm{cyl\_eq}') : \Pi\, y : B.\, \Pi\, w : \mathbf{Cyl}_f\, y.\, P\, y\, w}$$

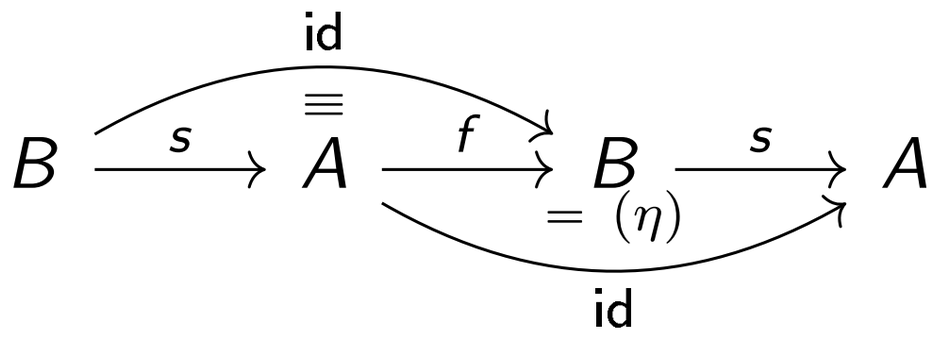
(C,AF)-WFS



Cofibrations:



Surjective Equivalences (AF):



$$+\ \mathsf{ap}\ \ f\ \ \eta_x\ \equiv\ \mathsf{refl}_{f(x)}$$

(C,AF)-WFS

To get a WFS, we need to have the following ***strict*** equalities.

$$\mathbf{cyl\_ind}(P, \mathrm{top}', \mathrm{base}', \mathrm{cyl\_eq}', f\ x, \mathbf{top}\ x) \equiv_{\alpha\beta\eta} \mathrm{top}'\ x$$

$$\mathbf{cyl\_ind}(P, \mathrm{top}', \mathrm{base}', \mathrm{cyl\_eq}', y, \mathbf{base}\ y) \equiv_{\alpha\beta\eta} \mathrm{base}'\ y$$

$$\mathbf{ap}\ \mathbf{cyl\_ind}(P, \mathrm{top}', \mathrm{base}', \mathrm{cyl\_eq}', f\ x)\ (\mathbf{cyl\_eq}_f\ x) \equiv \mathrm{cyl\_eq}'\ x$$

(C,AF)-WFS

To get a WFS, we need to have the following ***strict*** equalities.

$$\mathbf{cyl\_ind}(P, \mathrm{top}', \mathrm{base}', \mathrm{cyl\_eq}', f\ x, \mathbf{top}\ x) \equiv_{\alpha\beta\eta} \mathrm{top}'\ x$$

$$\mathbf{cyl\_ind}(P, \mathrm{top}', \mathrm{base}', \mathrm{cyl\_eq}', y, \mathbf{base}\ y) \equiv_{\alpha\beta\eta} \mathrm{base}'\ y$$

$$\mathbf{ap}\ \mathbf{cyl\_ind}(P, \mathrm{top}', \mathrm{base}', \mathrm{cyl\_eq}', f\ x)\ (\mathbf{cyl\_eq}_f\ x) \equiv \mathrm{cyl\_eq}'\ x$$

The last equality is usually assume to be ***up-to homotopy***, we need to investigate more to check that point.

Model Structure on Fibrant Types

▸ Theorem 1. There is a model structure on $\mathcal{U}_i^F$ with the weak equivalences, fibrations and cofibrations as previously defined.

Part IV. Model Structure on all types

Model Structure on Types

Mike Shulman and Paolo Capriotti have already noticed independently that the notion of fibrant replacement is ***inconsistent with HTS.***

Model Structure on Types

Fibrant replacement

It is natural to wonder whether we can have a "fibrant replacement" type former which makes non-fibrant types into fibrant ones. However, surprisingly, this is actually inconsistent, essentially because it cannot be made to respect substitution.

Suppose we had a type forming rule

$$\frac{\Gamma \vdash A\,\mathrm{Type}}{\Gamma \vdash RA\,\mathrm{Fib}}$$

with introduction rule

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash ra : RA}$$

and elimination rule

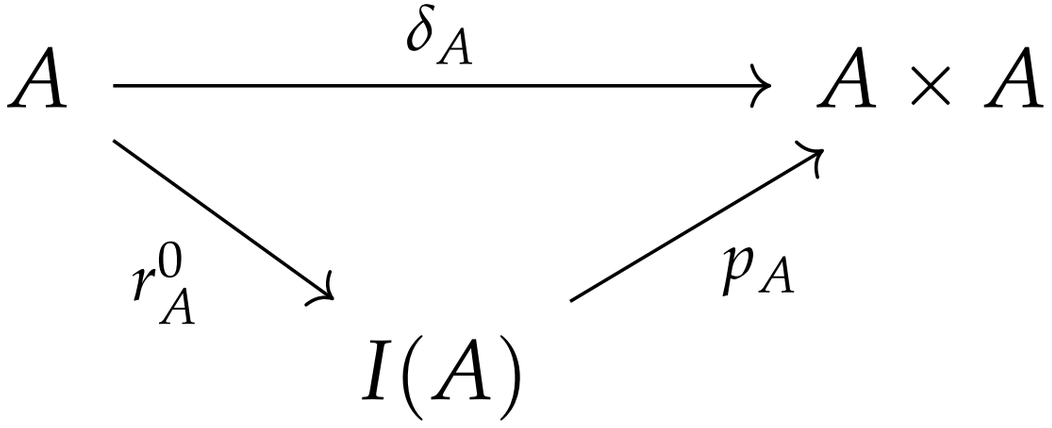
$$\frac{\Gamma, (x : RA) \vdash T\,\mathrm{Fib} \qquad \Gamma, (a : A) \vdash t : T[x :\!= ra]}{\Gamma, (x : RA) \vdash \mathrm{Relim}(a.\, t, x) : T}.$$

M. Shulman post in ncatlab.org

Model Structure on Types

Here, we generalize a bit this statement.

Identity Path induced by a Model Structure



$$e : p_A \circ r^0_A \equiv \delta_A$$

$$\mathbf{Id}_A \;:=\; \lambda\, x,\, y.\; \Sigma_{w:I(A)} p_A\, w \equiv (x,\, y)$$

The lifiting property of the wfs allows to derive the J eliminator.

Model Structure on Types

Impossibility Result: No model structure on HTS exists, for which

for all $x, P\,x$ is fibrant, then $\pi_1 : \Sigma_x P\,x \to A$ is a fibration

and

Id is equivalent to =.

Model Structure on Types

Idea of the Proof:

The proof is similar to the proof of Capriotti and goes by proving that Id satisfies UIP, so it can not be equivalent to =.

A Context-Dependent Notion of Fibrancy

We would like to suggest a variant of HTS which may solve the issue.

A Context-Dependent Notion of Fibrancy

We would like to suggest a variant of HTS which may solve the issue.

Disclaimer: We have no model for this new system for the moment.

A Context-Dependent Notion of Fibrancy

The idea is to have a new judgment for fibrancy

$$\Gamma \vdash (\Delta\,;\, A)\ \mathbf{Fib}$$

which says that in context Γ the family $A : \Delta \to \mathcal{U}_i$ is ***uniformly fibrant.***

A Context-Dependent Notion of Fibrancy

The idea is to have a new judgment for fibrancy

$$\Gamma \vdash (\Delta\,;\, A)\ \mathbf{Fib}$$

which says that in context Γ the family $A : \Delta \to \mathcal{U}_i$ is ***uniformly fibrant.***

But A may not be uniformly fibrant with respect to Γ !

A Context-Dependent Notion of Fibrancy

The rule of fibrancy are modified accordingly, for instance

$$\frac{\Gamma, \Delta, x : A \vdash B : \mathcal{U}_i \qquad \Gamma \vdash (\Delta, x : A\ ;\ B)\ \mathbf{Fib}}{\Gamma \vdash (\Delta\ ;\ \Pi\, x : A.\ B)\ \mathbf{Fib}}$$

A Context-Dependent Notion of Fibrancy

The fibrant replacement provides only ***pointwise*** fibrant families (i.e., in the empty context):

$$\frac{\Gamma \vdash A : \mathcal{U}_i}{\Gamma \vdash (\,.\,;\,\overline{A})\ \mathbf{Fib}}$$

Hack in Coq

A similar—although trickier—hack works to emulate this new system in Coq.

Weak Equivalences Revisited

a function $f : A \to B$ is weak-equivalence if $\overline{f} : \overline{A} \to \overline{B}$ is a type equivalence

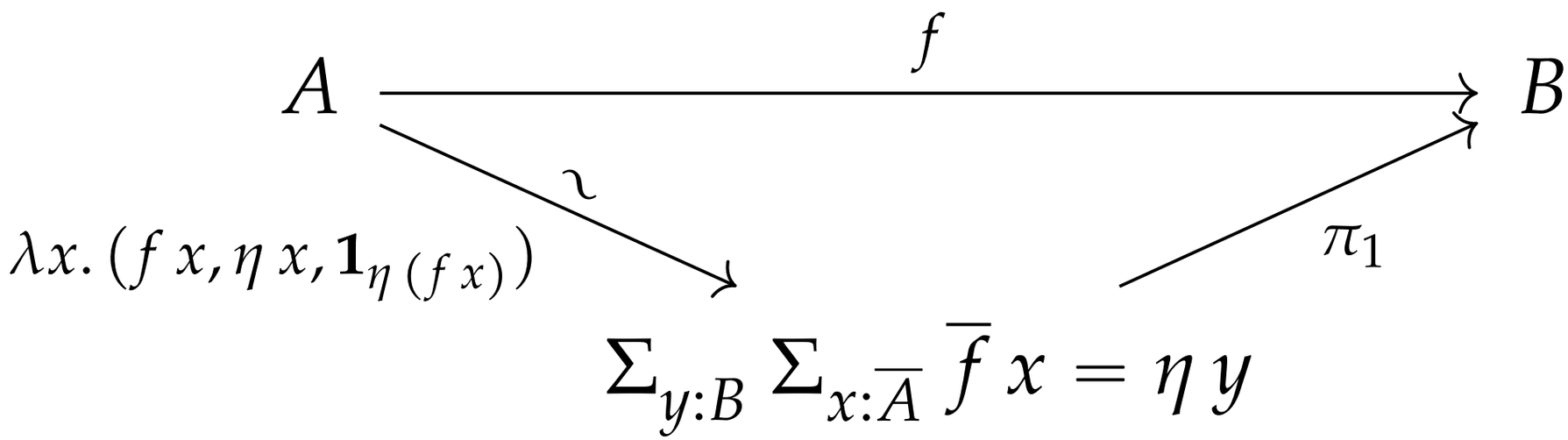
Similar to the definition of weak equivalences in the simplicial model

- The **weak equivalences** W are **weak homotopy equivalences**, i.e., morphisms whose geometric realization is a weak homotopy equivalence of topological spaces.

From ncatlab.org

(AC/F) WFS revisited

The factorisation of a function f as an AC/F now makes use of the fibrant replacement.



Model Structure on Type

Theorem: there is a model structure on Type in this new system.

Doggy bag of the talk

1. In HTS, we can define model structures and show that there is a model structure on fibrant types.

2. Among mild assumptions, there is no model structure on all types in HTS.

3. We propose an (unproven) refinement of HTS that admits a model structure on all type based on the notion of ***uniform fibrancy***.

4. This model structure can be used to justify/compute homopical limits and colimits.