Higher Pro-Arrows: Towards a Model for Naturality Pretype Theory

Andreas Nuyts

DistriNet
KU LEUVEN

HoTTEST
May 2, 2024

https://anuyts.github.io/files/2024/natpt-hottest-pres.pdf

Introduction

Why I care

Not out of an intrinsic interest in

- (directed) algebraic topology,
- synthetic (∞, ∞)-category theory.

Consequences

- Types stratified by finite dimensions. (Cf. Haskell but less weird.)
- I'm not afraid of strict equality. I am afraid of coherence obligations.
- I don't mind if my model doesn't present spaces. But I want it to compute!
- Factorization systems are not my native language.

I want better languages for **verified functional programming**!

Programs should be categorically structured.

With native support for relations/morphisms/isomorphisms:

- Parametricity for free!
- Functoriality for free!
- Naturality for free!
- Variance of dependent multi-argument functions sorted out for free!

Why I care

Not out of an intrinsic interest in

- (directed) algebraic topology,
- synthetic (∞, ∞)-category theory.

Consequences

- Types stratified by finite dimensions. (Cf. Haskell but less weird.)
- I'm not afraid of strict equality. I am afraid of coherence obligations.
- I don't mind if my model doesn't present spaces. But I want it to compute!
- Factorization systems are not my native language.

I want better languages for **verified functional programming**!

Programs should be categorically structured.

With native support for relations/morphisms/isomorphisms:

- Parametricity for free!
- Functoriality for free!
- Naturality for free!
- Variance of dependent multi-argument functions sorted out for free!

Why I care

Not out of an intrinsic interest in

- (directed) algebraic topology,
- synthetic (∞, ∞)-category theory.

Consequences

- Types stratified by finite dimensions. (Cf. Haskell but less weird.)
- I'm not afraid of strict equality. I am afraid of coherence obligations.
- I don't mind if my model doesn't present spaces. But I want it to compute!
- Factorization systems are not my native language.

I want better languages for **verified functional programming**!

Programs should be categorically structured.

With native support for relations/morphisms/isomorphisms:

- Parametricity for free!
- Functoriality for free!
- Naturality for free!
- Variance of dependent multi-argument functions sorted out for free!

Why I care

Not out of an intrinsic interest in

- (directed) algebraic topology,
- synthetic (∞, ∞)-category theory.

Consequences

- Types stratified by finite dimensions.
  (Cf. Haskell but less weird.)
- I'm not afraid of strict equality.
  I am afraid of coherence obligations.
- I don't mind if my model doesn't present spaces. But I want it to compute!
- Factorization systems are not my native language.

I want better languages for **verified functional programming**!

Programs should be categorically structured.

With native support for relations/morphisms/isomorphisms:

- Parametricity for free!
- Functoriality for free!
  ... and not just for $\mathrm{Type} \to \mathrm{Type}$
- Naturality for free!
- Variance of dependent multi-argument functions sorted out for free!

Why I care

Not out of an intrinsic interest in

- (directed) algebraic topology,
- synthetic (∞, ∞)-category theory.

Consequences

- Types stratified by finite dimensions. (Cf. Haskell but less weird.)
- I'm not afraid of strict equality. I am afraid of coherence obligations.
- I don't mind if my model doesn't present spaces. But I want it to compute!
- Factorization systems are not my native language.

I want better languages for **verified functional programming**!

Programs should be categorically structured.

With native support for relations/morphisms/isomorphisms:

- Parametricity for free!
- Functoriality for free!
  ... and not just for Type $\to$ Type
- Naturality for free!
- Variance of dependent multi-argument functions sorted out for free!

Why I care

Not out of an intrinsic interest in

- (directed) algebraic topology,
- synthetic (∞, ∞)-category theory.

Consequences

- Types stratified by finite dimensions. (Cf. Haskell but less weird.)
- I'm not afraid of strict equality. I am afraid of coherence obligations.
- I don't mind if my model doesn't present spaces. But I want it to compute!
- Factorization systems are not my native language.

I want better languages for **verified functional programming**!

Programs should be categorically structured.

With native support for relations/morphisms/isomorphisms:

- Parametricity for free!
- Functoriality for free!
  . . . and not just for Type $\rightarrow$ Type
- Naturality for free!
- Variance of dependent multi-argument functions sorted out for free!

Why I care

Not out of an intrinsic interest in

- (directed) algebraic topology,
- synthetic (∞, ∞)-category theory.

Consequences

- Types stratified by finite dimensions. (Cf. Haskell but less weird.)
- I'm not afraid of strict equality. I am afraid of coherence obligations.
- I don't mind if my model doesn't present spaces. But I want it to compute!
- Factorization systems are not my native language.

I want better languages for **verified functional programming**!

Programs should be categorically structured.

With native support for relations/morphisms/isomorphisms:

- Parametricity for free!
- Functoriality for free!
  …and not just for Type $\to$ Type
- Naturality for free!
- Variance of dependent multi-argument functions sorted out for free!

Why I care

Not out of an intrinsic interest in

- (directed) algebraic topology,
- synthetic (∞, ∞)-category theory.

Consequences

- Types stratified by finite dimensions.
  (Cf. Haskell but less weird.)
- I'm not afraid of strict equality.
  I am afraid of coherence obligations.
- I don't mind if my model doesn't present spaces. But I want it to **compute**!
- Factorization systems are not my native language.

I want better languages for **verified functional programming**!

Programs should be categorically structured.

With native support for relations/morphisms/isomorphisms:

- Parametricity for free!
- Functoriality for free!
  . . . and not just for Type $\to$ Type
- Naturality for free!
- Variance of dependent multi-argument functions sorted out for free!

Why I care

Not out of an intrinsic interest in

- (directed) algebraic topology,
- synthetic (∞, ∞)-category theory.

Consequences

- Types stratified by finite dimensions.
  (Cf. Haskell but less weird.)
- I'm not afraid of strict equality.
  I am afraid of coherence obligations.
- I don't mind if my model doesn't present spaces. But I want it to **compute**!
- Factorization systems are not my native language.

I want better languages for **verified functional programming**!

Programs should be categorically structured.

With native support for relations/morphisms/isomorphisms:

- Parametricity for free!
- Functoriality for free!
  ... and not just for Type $\to$ Type
- Naturality for free!
- Variance of dependent multi-argument functions sorted out for free!

Why I care

Not out of an intrinsic interest in

- (directed) algebraic topology,
- synthetic (∞, ∞)-category theory.

Consequences

- Types stratified by finite dimensions.
  (Cf. Haskell but less weird.)
- I'm not afraid of strict equality.
  I am afraid of coherence obligations.
- I don't mind if my model doesn't present spaces. But I want it to **compute**!
- Factorization systems are not my native language.

I want better languages for **verified functional programming**!

Programs should be categorically structured.

With native support for relations/morphisms/isomorphisms:

- Parametricity for free!
- Functoriality for free!
  ... and not just for Type $\to$ Type
- Naturality for free!
- Variance of dependent multi-argument functions sorted out for free!

Why I care

Not out of an intrinsic interest in

- (directed) algebraic topology,
- synthetic (∞, ∞)-category theory.

Consequences

- Types stratified by finite dimensions.
  (Cf. Haskell but less weird.)
- I'm not afraid of strict equality.
  I am afraid of coherence obligations.
- I don't mind if my model doesn't present spaces. But I want it to **compute**!
- Factorization systems are not my native language.

I want better languages for **verified functional programming**!

Programs should be categorically structured.

With native support for relations/morphisms/isomorphisms:

- Parametricity for free!
- Functoriality for free!
  ...and not just for Type $\to$ Type
- Naturality for free!
- Variance of dependent multi-argument functions sorted out for free!

So how is **Directed TT** relevant to
verified functional programming?

An example problem

$$
\begin{tikzcd}
 & & M & N & & \mathrm{Monad} \\
A \arrow[d, "f"'] & \mathrm{List}\,A & \mathrm{WriterT}\,(\mathrm{List}\,A)\,M & \mathrm{WriterT}\,(\mathrm{List}\,A)\,N & & \\
B & \mathrm{List}\,B & \mathrm{WriterT}\,(\mathrm{List}\,B)\,M & \mathrm{WriterT}\,(\mathrm{List}\,B)\,N & & \\
\pitchfork & & & & \mathrm{Monad} & \\
\mathrm{Type} & \mathrm{Monoid} & & & & \times
\end{tikzcd}
$$

$$
\begin{tikzcd}
 & & M & N & & \mathrm{Monad} \\
A \arrow[d, "f"'] & \mathrm{List}\,A & \mathrm{WriterT}\,(\mathrm{List}\,A)\,M & \mathrm{WriterT}\,(\mathrm{List}\,A)\,N & & \\
B & \mathrm{List}\,B & \mathrm{WriterT}\,(\mathrm{List}\,B)\,M & \mathrm{WriterT}\,(\mathrm{List}\,B)\,N & & \\
\pitchfork & & & & \mathrm{Monad} & \\
\mathrm{Type} \arrow[r, "\mathrm{List}"] & \mathrm{Monoid} & & & & \times
\end{tikzcd}
$$

$$
\begin{tikzcd}
 & & M & N & \mathrm{Monad} \\
A \arrow[d, "f"'] & \mathrm{List}\,A \arrow[d, "\mathrm{List}\,f"'] & \mathrm{WriterT}\,(\mathrm{List}\,A)\,M & \mathrm{WriterT}\,(\mathrm{List}\,A)\,N & \\
B & \mathrm{List}\,B & \mathrm{WriterT}\,(\mathrm{List}\,B)\,M & \mathrm{WriterT}\,(\mathrm{List}\,B)\,N & \\
\pitchfork & \pitchfork & & \mathrm{Monad} & \\
\mathrm{Type} \arrow[r, "\mathrm{List}"] & \mathrm{Monoid} & & & \times
\end{tikzcd}
$$

$$M \xrightarrow{g} N \in \mathsf{Monad}$$

$$\begin{tikzcd}
A \arrow[d, "f"'] & \mathsf{List}\,A \arrow[d, "\mathsf{List}\,f"'] & \mathsf{WriterT}\,(\mathsf{List}\,A)\,M & \mathsf{WriterT}\,(\mathsf{List}\,A)\,N \\
B & \mathsf{List}\,B & \mathsf{WriterT}\,(\mathsf{List}\,B)\,M & \mathsf{WriterT}\,(\mathsf{List}\,B)\,N \\
\pitchfork & \pitchfork & & \mathsf{Monad}
\end{tikzcd}$$

$$\mathsf{Type} \xrightarrow{\mathsf{List}} \mathsf{Monoid} \qquad \times$$

$$
\begin{tikzcd}
 & & M \arrow[r, "g"] & N & \in & \mathrm{Monad} \arrow[dddd, no head] \\
A \arrow[d, "f"'] & \mathrm{List}\,A \arrow[d, "\mathrm{List}\,f"'] & \mathrm{WriterT}\,(\mathrm{List}\,A)\,M & \mathrm{WriterT}\,(\mathrm{List}\,A)\,N & & \\
B & \mathrm{List}\,B & \mathrm{WriterT}\,(\mathrm{List}\,B)\,M & \mathrm{WriterT}\,(\mathrm{List}\,B)\,N & & \\
\pitchfork & \pitchfork & & & \mathrm{Monad} & \\
\mathrm{Type} \arrow[r, "\mathrm{List}"] & \mathrm{Monoid} \arrow[rrrr, no head] & & & & \times \arrow[ul, "\mathrm{WriterT}"']
\end{tikzcd}
$$

$$
\begin{tikzcd}
 & & M \arrow[r, "g"] & N & \in & \mathrm{Monad} \arrow[dddd, no head] \\
A \arrow[d, "f"'] & \mathrm{List}\,A \arrow[d, "\mathrm{List}\,f"'] & \mathrm{WriterT}\,(\mathrm{List}\,A)\,M \arrow[r, bend left, "\mathrm{WriterT}\,(\mathrm{List}\,A)\,g"] \arrow[d, "\mathrm{WriterT}\,(\mathrm{List}\,f)\,M"'] & \mathrm{WriterT}\,(\mathrm{List}\,A)\,N \arrow[d, "\mathrm{WriterT}\,(\mathrm{List}\,f)\,N"'] & & \\
B & \mathrm{List}\,B & \mathrm{WriterT}\,(\mathrm{List}\,B)\,M \arrow[r, bend right, "\mathrm{WriterT}\,(\mathrm{List}\,B)\,g"'] & \mathrm{WriterT}\,(\mathrm{List}\,B)\,N \arrow[dr, phantom, "\in"] & & \\
\rotatebox{90}{$\in$} & \rotatebox{90}{$\in$} & & & \mathrm{Monad} & \\
\mathrm{Type} \arrow[r, "\mathrm{List}"] & \mathrm{Monoid} \arrow[rrrr, no head] & & & & \times \arrow[ul, "\mathrm{WriterT}"']
\end{tikzcd}
$$

$$
\begin{tikzcd}
 & & M \arrow[r, "g"] & N & \in & \mathrm{Monad} \arrow[dddd, dash] \\
A \arrow[d, "f"'] & \mathrm{List}\,A \arrow[d, "\mathrm{List}\,f"'] & \mathrm{WriterT}\,(\mathrm{List}\,A)\,M \arrow[r, bend left, "\mathrm{WriterT}\,(\mathrm{List}\,A)\,g"] \arrow[d, "\mathrm{WriterT}\,(\mathrm{List}\,f)\,M"'] \arrow[dr, phantom, "\textbf{?}"] & \mathrm{WriterT}\,(\mathrm{List}\,A)\,N \arrow[d, "\mathrm{WriterT}\,(\mathrm{List}\,f)\,N"'] & & \\
B & \mathrm{List}\,B & \mathrm{WriterT}\,(\mathrm{List}\,B)\,M \arrow[r, bend right, "\mathrm{WriterT}\,(\mathrm{List}\,B)\,g"'] & \mathrm{WriterT}\,(\mathrm{List}\,B)\,N & \rotatebox{-60}{$\in$} & \\
\pitchfork & \pitchfork & & & \mathrm{Monad} & \\
\mathrm{Type} \arrow[r, "\mathrm{List}"] & \mathrm{Monoid} \arrow[rrrr, dash] & & & & \times \arrow[ul, "\mathrm{WriterT}"']
\end{tikzcd}
$$

In plain DTT

Functoriality of List : Type $\to$ Monoid:

- Object action: $(\mathrm{List}\, A, [], ++)$
- Functorial action:
  - $\mathrm{List}\, f : \mathrm{List}\, A \to \mathrm{List}\, B$ **(by recursion)**
  - $\mathrm{List}\, f$ is a monoid morphism:
    - $\mathrm{List}\, f$ preserves $[]$ (trivial)
    - $\mathrm{List}\, f$ preserves $++$ **(by induction)**

  $+$ functor laws **(by induction)**

Functoriality of

WriterT : Monoid $\to$ MonadTrans

- Object action: $\mathrm{WriterT}\, W \in \mathrm{MonadTrans}$
  - Object action: $\mathrm{WriterT}\, W\, M \in \mathrm{Monad}$
    - Object action: Define $\mathrm{WriterT}\, W\, M\, A$
    - Functorial action $\mathrm{WriterT}\, W\, M\, f$

      $+$ functor laws
    - return & bind $+$ naturality

. . . Object action: $\mathrm{WriterT}\, W \in \mathrm{MonadTrans}$

  - Functorial action $\mathrm{WriterT}\, W\, g$
    - Respects return & bind

    $+$ functor laws
  - lift : $M \to \mathrm{WriterT}\, W\, M$ $+$ naturality
    - Respects return & bind
- Functorial action:

  $\mathrm{WriterT}\, h : \mathrm{WriterT}\, V \to \mathrm{WriterT}\, W$
  - $\mathrm{WriterT}\, h\, M\, A$
    - Respects return, bind & lift
  - naturality w.r.t. A
  - naturality w.r.t. M

  $+$ functor laws

In plain DTT

Functoriality of List : Type $\to$ Monoid:

- Object action: $(\mathrm{List}\,A, [], ++)$
- Functorial action:
  - $\mathrm{List}\,f : \mathrm{List}\,A \to \mathrm{List}\,B$ **(by recursion)**
  - $\mathrm{List}\,f$ is a monoid morphism:
    - $\mathrm{List}\,f$ preserves $[]$ (trivial)
    - $\mathrm{List}\,f$ preserves $++$ **(by induction)**

  + functor laws **(by induction)**

Functoriality of WriterT : Monoid $\to$ MonadTrans

- Object action: $\mathrm{WriterT}\,W \in \mathrm{MonadTrans}$
  - Object action: $\mathrm{WriterT}\,W\,M \in \mathrm{Monad}$
    - Object action: Define $\mathrm{WriterT}\,W\,M\,A$
    - Functorial action $\mathrm{WriterT}\,W\,M\,f$

      + functor laws
    - return & bind + naturality

… Object action: $\mathrm{WriterT}\,W \in \mathrm{MonadTrans}$

- Functorial action $\mathrm{WriterT}\,W\,g$
  - Respects return & bind

  + functor laws
- lift : $M \to \mathrm{WriterT}\,W\,M$ + naturality
  - Respects return & bind

- Functorial action: $\mathrm{WriterT}\,h : \mathrm{WriterT}\,V \to \mathrm{WriterT}\,W$
  - $\mathrm{WriterT}\,h\,M\,A$
    - Respects return, bind & lift
  - naturality w.r.t. A
  - naturality w.r.t. M

  + functor laws

In plain DTT

Functoriality of List : Type $\to$ Monoid:

- Object action: $(\mathrm{List}\,A, [], +\!+)$
- Functorial action:
  - $\mathrm{List}\,f : \mathrm{List}\,A \to \mathrm{List}\,B$ **(by recursion)**
  - $\mathrm{List}\,f$ is a monoid morphism:
    - $\mathrm{List}\,f$ preserves $[]$ (trivial)
    - $\mathrm{List}\,f$ preserves $+\!+$ **(by induction)**

  + functor laws **(by induction)**

Functoriality of
WriterT : Monoid $\to$ MonadTrans

- Object action: $\mathrm{WriterT}\,W \in \mathrm{MonadTrans}$
  - Object action: $\mathrm{WriterT}\,W\,M \in \mathrm{Monad}$
    - Object action: Define $\mathrm{WriterT}\,W\,M\,A$
    - Functorial action $\mathrm{WriterT}\,W\,M\,f$

      + functor laws
    - return & bind + naturality

... Object action: $\mathrm{WriterT}\,W \in \mathrm{MonadTrans}$

  - Functorial action $\mathrm{WriterT}\,W\,g$
    - Respects return & bind

    + functor laws
  - lift : $M \to \mathrm{WriterT}\,W\,M$ + naturality
    - Respects return & bind
- Functorial action:
  $\mathrm{WriterT}\,h : \mathrm{WriterT}\,V \to \mathrm{WriterT}\,W$
  - $\mathrm{WriterT}\,h\,M\,A$
    - Respects return, bind & lift
  - naturality w.r.t. A
  - naturality w.r.t. M

  + functor laws

In parametric DTT

Functoriality of $\mathrm{List} : \mathrm{Type} \to \mathrm{Monoid}$:

- Object action: $(\mathrm{List}\, A, [], ++)$
- Functorial action:
  - $\mathrm{List}\, f : \mathrm{List}\, A \to \mathrm{List}\, B$ **(by recursion)**
  - $\mathrm{List}\, f$ is a monoid morphism:
    - $\mathrm{List}\, f$ preserves $[]$ (trivial)
    - $\mathrm{List}\, f$ preserves $++$ **(by induction)**

  + functor laws **(by induction)**

Functoriality of
$\mathrm{WriterT} : \mathrm{Monoid} \to \mathrm{MonadTrans}$

- Object action: $\mathrm{WriterT}\, W \in \mathrm{MonadTrans}$
  - Object action: $\mathrm{WriterT}\, W\, M \in \mathrm{Monad}$
    - Object action: Define $\mathrm{WriterT}\, W\, M\, A$
    - Functorial action $\mathrm{WriterT}\, W\, M\, f$
      + functor laws
    - return & bind + naturality

… Object action: $\mathrm{WriterT}\, W \in \mathrm{MonadTrans}$
  - Functorial action $\mathrm{WriterT}\, W\, g$
    - Respects return & bind

    + functor laws
  - $\mathrm{lift} : M \to \mathrm{WriterT}\, W\, M$ + naturality
    - Respects return & bind
- Functorial action:
  $\mathrm{WriterT}\, h : \mathrm{WriterT}\, V \to \mathrm{WriterT}\, W$
  - $\mathrm{WriterT}\, h\, M\, A$
    - Respects return, bind & lift
  - naturality w.r.t. A
  - naturality w.r.t. M

  + functor laws

$$
\begin{tikzcd}
 & & M \arrow[rr, "g", "\cong"'] & & N & \in & \mathrm{Monad} \arrow[ddd, dash] \\
A \arrow[d, "f"', "\cong"] & \mathrm{List}\,A \arrow[d, "\mathrm{List}\,f"', "\cong"] & \mathrm{WriterT}\,(\mathrm{List}\,A)\,M \arrow[rr, bend left, "\mathrm{WriterT}\,(\mathrm{List}\,A)\,g", "\cong"'] \arrow[d, "\mathrm{WriterT}\,(\mathrm{List}\,f)\,M", "\cong"'] \arrow[drr, phantom, "?"] & & \mathrm{WriterT}\,(\mathrm{List}\,A)\,N \arrow[d, "\mathrm{WriterT}\,(\mathrm{List}\,f)\,N", "\cong"'] & & \\
B & \mathrm{List}\,B & \mathrm{WriterT}\,(\mathrm{List}\,B)\,M \arrow[rr, bend right, "\mathrm{WriterT}\,(\mathrm{List}\,B)\,g"', "\cong"] & & \mathrm{WriterT}\,(\mathrm{List}\,B)\,N & \mathrm{Monad} & \\
\mathrm{Type} \arrow[r, "\mathrm{List}"] & \mathrm{Monoid} \arrow[rrrrr, dash] & & & & & \times \arrow[ul, "\mathrm{WriterT}"]
\end{tikzcd}
$$

$$
\begin{tikzcd}
 & & M \arrow[r, equal] & N & \in & \mathsf{Monad} \arrow[dddd, no head] \\
A \arrow[d, equal] & \mathsf{List}\,A \arrow[d, equal] & \mathsf{WriterT}\,(\mathsf{List}\,A)\,M \arrow[r, equal, bend left] \arrow[d, equal] \arrow[dr, phantom, "\color{purple}\textbf{?}"] & \mathsf{WriterT}\,(\mathsf{List}\,A)\,N \arrow[d, equal] & & \\
B \arrow[dd, phantom, "\rotatebox[origin=c]{-90}{$\in$}"] & \mathsf{List}\,B \arrow[dd, phantom, "\rotatebox[origin=c]{-90}{$\in$}"] & \mathsf{WriterT}\,(\mathsf{List}\,B)\,M \arrow[r, equal, bend right] & \mathsf{WriterT}\,(\mathsf{List}\,B)\,N \arrow[dr, phantom, "\rotatebox[origin=c]{-45}{$\in$}"] & & \\
 & & & & \mathsf{Monad} & \\
\mathsf{Type} \arrow[r, "\mathsf{List}"] & \mathsf{Monoid} \arrow[rrrr, no head] & & & & \times \arrow[ul, "\mathsf{WriterT}"']
\end{tikzcd}
$$

In HoTT (assuming f, g and $h = \mathsf{List}\, f$ are isos)

Functoriality of List : Type $\to$ Monoid:

- Object action: $(\mathsf{List}\, A, [], ++)$
- 🎁 Functorial action:
  - 🎁 $\mathsf{List}\, f : \mathsf{List}\, A \cong \mathsf{List}\, B$ ~~(by recursion)~~
  - 🎁 $\mathsf{List}\, f$ is a monoid morphism:
    - 🎁 $\mathsf{List}\, f$ preserves $[]$ (trivial)
    - 🎁 $\mathsf{List}\, f$ preserves $++$ ~~(by ind.)~~

  \+ 🎁 functor laws ~~(by induction)~~

Functoriality of
WriterT : Monoid $\to$ MonadTrans

- Object action: $\mathsf{WriterT}\, W \in \mathsf{MonadTrans}$
  - Object action: $\mathsf{WriterT}\, W\, M \in \mathsf{Monad}$
    - Object action: Define $\mathsf{WriterT}\, W\, M\, A$
    - 🎁 Functorial action $\mathsf{WriterT}\, W\, M\, f$

      \+ 🎁 functor laws
    - return & bind + 🎁 naturality

. . . Object action: $\mathsf{WriterT}\, W \in \mathsf{MonadTrans}$

  - 🎁 Functorial action $\mathsf{WriterT}\, W\, g$
    - 🎁 Respects return & bind

    \+ 🎁 functor laws
  - lift : $M \to \mathsf{WriterT}\, W\, M$ + 🎁 naturality
    - Respects return & bind
- 🎁 Functorial action:
  $\mathsf{WriterT}\, h : \mathsf{WriterT}\, V \cong \mathsf{WriterT}\, W$
  - 🎁 $\mathsf{WriterT}\, h\, M\, A$
    - 🎁 Respects return, bind & lift
  - 🎁 naturality w.r.t. A
  - 🎁 naturality w.r.t. M

  \+ 🎁 functor laws

In Naturality TT

Functoriality of List : Type $\to$ Monoid:

- Object action: $(\mathrm{List}\,A, [], ++)$
- 🎁 Functorial action:
  - 🎁 List f : List $A \to$ List B ~~(by recursion)~~
  - 🎁 List f is a monoid morphism:
    - 🎁 List f preserves [] (trivial)
    - 🎁 List f preserves $++$ ~~(by ind.)~~

  \+ 🎁 functor laws ~~(by induction)~~

Functoriality of

WriterT : Monoid $\to$ MonadTrans

- Object action: WriterT $W \in$ MonadTrans
  - Object action: WriterT $W\,M \in$ Monad
    - Object action: Define WriterT $W\,M\,A$
    - 🎁 Functorial action WriterT $W\,M\,f$

      \+ 🎁 functor laws
    - return & bind + 🎁 naturality

... Object action: WriterT $W \in$ MonadTrans

  - 🎁 Functorial action WriterT $W\,g$
    - 🎁 Respects return & bind

    \+ 🎁 functor laws
  - lift : $M \to$ WriterT $W\,M$ + 🎁 naturality
    - Respects return & bind
- 🎁 Functorial action:

  WriterT h : WriterT $V \to$ WriterT W
  - 🎁 WriterT $h\,M\,A$
    - 🎁 Respects return, bind & lift
  - 🎁 naturality w.r.t. A
  - 🎁 naturality w.r.t. M

  \+ 🎁 functor laws

Variance and modalities

WriterT $W\,M\,A := M(A \times W)$ is **covariant** w.r.t.

- W : Monoid
- M : Monad
- A : Type

ReaderT $R\,M\,A := R \to MA$ is **contravariant** w.r.t.

- R : Type

return : $A \to$ WriterT $W\,M\,A$ is **natural** w.r.t.

- W : Monoid
- M : Monad
- A : Type

Ignoring variance

- HoTT: only consider **isomorphisms**
  ☹ Not everything is an isomorphism.
- Param'ty: **relations**, not morphisms
  ☹ Don't know how to compute fmap.

Naturality TT

- Preserve isomorphisms
- Preserve relations
- Keep track of action on morphisms

Hence:

- Use functoriality/naturality when possible
- Use HoTT when applicable
- Use param'ty when necessary

Variance and modalities

WriterT $W\,M\,A := M(A \times W)$ is **covariant** w.r.t.

- W : Monoid
- M : Monad
- A : Type

ReaderT $R\,M\,A := R \to MA$ is **contravariant** w.r.t.

- R : Type

return : $A \to$ WriterT $W\,M\,A$ is **natural** w.r.t.

- W : Monoid
- M : Monad
- A : Type

Ignoring variance

- HoTT: only consider **isomorphisms**
  ☹ Not everything is an isomorphism.
- Param'ty: **relations**, not morphisms
  ☹ Don't know how to compute fmap.

Naturality TT

- Preserve isomorphisms
- Preserve relations
- Keep track of action on morphisms

Hence:

- Use functoriality/naturality when possible
- Use HoTT when applicable
- Use param'ty when necessary

Variance and modalities

$\text{WriterT}\, W\, M\, A := M(A \times W)$ is **covariant** w.r.t.

- W : Monoid
- M : Monad
- A : Type

$\text{ReaderT}\, R\, M\, A := R \to M A$ is **contravariant** w.r.t.

- R : Type

$\text{return} : A \to \text{WriterT}\, W\, M\, A$ is **natural** w.r.t.

- W : Monoid
- M : Monad
- A : Type

Ignoring variance

- HoTT: only consider **isomorphisms**
  ☹ Not everything is an isomorphism.
- Param'ty: **relations**, not morphisms
  ☹ Don't know how to compute fmap.

Naturality TT

- Preserve isomorphisms
- Preserve relations
- Keep track of action on morphisms

Hence:

- Use functoriality/naturality when possible
- Use HoTT when applicable
- Use param'ty when necessary

Variance and modalities

$\mathrm{WriterT}\,W\,M\,A := M(A \times W)$ is **covariant** w.r.t.

- W : Monoid
- M : Monad
- A : Type

$\mathrm{ReaderT}\,R\,M\,A := R \to MA$ is **contravariant** w.r.t.

- R : Type

$\mathrm{return} : A \to \mathrm{WriterT}\,W\,M\,A$ is **natural** w.r.t.

- W : Monoid
- M : Monad
- A : Type

Ignoring variance

- HoTT: only consider **isomorphisms**
  ☹ Not everything is an isomorphism.
- Param'ty: **relations**, not morphisms
  ☹ Don't know how to compute fmap.

Naturality TT

- Preserve isomorphisms
- Preserve relations
- Keep track of action on morphisms

Hence:

- Use functoriality/naturality when possible
- Use HoTT when applicable
- Use param'ty when necessary

Variance and modalities

WriterT $W\,M\,A := M(A \times W)$ is **covariant** w.r.t.

- W : Monoid
- M : Monad
- A : Type

ReaderT $R\,M\,A := R \to MA$ is **contravariant** w.r.t.

- R : Type

return : $A \to$ WriterT $W\,M\,A$ is **natural** w.r.t.

- W : Monoid
- M : Monad
- A : Type

Ignoring variance

- HoTT: only consider **isomorphisms**
  ☹ Not everything is an isomorphism.
- Param'ty: **relations**, not morphisms
  ☹ Don't know how to compute fmap.

Naturality TT

- Preserve isomorphisms
- Preserve relations
- Keep track of action on morphisms

Hence:

- Use functoriality/naturality when possible
- Use HoTT when applicable
- Use param'ty when necessary

Variance and modalities

WriterT $W\,M\,A := M(A \times W)$ is **covariant** w.r.t.

- W : Monoid
- M : Monad
- A : Type

ReaderT $R\,M\,A := R \to M A$ is **contravariant** w.r.t.

- R : Type

return : $A \to$ WriterT $W\,M\,A$ is **natural** w.r.t.

- W : Monoid
- M : Monad
- A : Type

Ignoring variance

- HoTT: only consider **isomorphisms**
  ☹ Not everything is an isomorphism.
- Param'ty: **relations**, not morphisms
  ☹ Don't know how to compute fmap.

Naturality TT

- Preserve isomorphisms
- Preserve relations
- Keep track of action on morphisms

Hence:

- Use functoriality/naturality when possible
- Use HoTT when applicable
- Use param'ty when necessary

Variance and modalities

$\mathrm{WriterT}\, W\, M\, A := M(A \times W)$ is **covariant** w.r.t.

- W : Monoid
- M : Monad
- A : Type

$\mathrm{ReaderT}\, R\, M\, A := R \to M A$ is **contravariant** w.r.t.

- R : Type

$\mathrm{return} : A \to \mathrm{WriterT}\, W\, M\, A$ is **natural** w.r.t.

- W : Monoid
- M : Monad
- A : Type

Ignoring variance

- HoTT: only consider **isomorphisms**
  ☹ Not everything is an isomorphism.
- Param'ty: **relations**, not morphisms
  ☹ Don't know how to compute fmap.

Naturality TT

- Preserve isomorphisms
- Preserve relations
- Keep track of action on morphisms

Hence:

- Use functoriality/naturality when possible
- Use HoTT when applicable
- Use param'ty when necessary

Pretypes: A Note on Fibrancy

A note on fibrancy

A **presheaf model** of DTT can account for:

- The **existence** of shapes
  (point, path, morphism, bridge, . . .)
- **Unary operations** on shapes (src, rfl)
- **Unary equations** on shapes
  ($\mathrm{src} \circ \mathrm{rfl} = \mathrm{id}$)

Fibrancy allows for:

- Other arities (composition, . . .)
- Specific geometries (transport, . . .)

HoTT	
Kan	Comp. of & transp. along paths
Directed	
functorial	Transport along morphisms
Segal	Composition of morphisms
Rezk	Isomorphism-path univalence
Param'ty	
discrete	Homog. bridges express equality
. . .	. . .

A note on fibrancy

A **presheaf model** of DTT can account for:

- The **existence** of shapes (point, path, morphism, bridge, ...)
- **Unary operations** on shapes (src, rfl)
- **Unary equations** on shapes ($\mathrm{src} \circ \mathrm{rfl} = \mathrm{id}$)

Fibrancy allows for:

- Other arities (composition, ...)
- Specific geometries (transport, ...)

HoTT	
Kan	Comp. of & transp. along paths
Directed	
functorial	Transport along morphisms
Segal	Composition of morphisms
Rezk	Isomorphism-path univalence
Param'ty	
discrete	Homog. bridges express equality
...	...

A note on fibrancy

A **presheaf model** of DTT can account for:

- The **existence** of shapes
  (point, path, morphism, bridge, . . .)
- **Unary operations** on shapes (src, rfl)
- **Unary equations** on shapes
  (src $\circ$ rfl $=$ id)

Fibrancy allows for:

- Other arities (composition, . . .)
- Specific geometries (transport, . . .)

HoTT	
Kan	Comp. of & transp. along paths
Directed	
functorial	Transport along morphisms
Segal	Composition of morphisms
Rezk	Isomorphism-path univalence
Param'ty	
discrete	Homog. bridges express equality
. . .	. . .

A note on fibrancy

A **presheaf model** of DTT can account for:

- The **existence** of shapes (point, path, morphism, bridge, . . .)
- **Unary operations** on shapes (src, rfl)
- **Unary equations** on shapes ($\mathrm{src} \circ \mathrm{rfl} = \mathrm{id}$)

Fibrancy allows for:

- Other arities (composition, . . .)
- Specific geometries (transport, . . .)

HoTT	
Kan	Comp. of & transp. along paths
Directed	
functorial	Transport along morphisms
Segal	Composition of morphisms
Rezk	Isomorphism-path univalence
Param'ty	
discrete	Homog. bridges express equality
. . .	. . .

A note on fibrancy

A **presheaf model** of DTT can account for:

- The **existence** of shapes
  (point, path, morphism, bridge, . . .)
- **Unary operations** on shapes (src, rfl)
- **Unary equations** on shapes
  $(\mathrm{src} \circ \mathrm{rfl} = \mathrm{id})$

Fibrancy allows for:

- Other arities (composition, . . .)
- Specific geometries (transport, . . .)

HoTT	
Kan	Comp. of & transp. along paths
Directed	
functorial	Transport along morphisms
Segal	Composition of morphisms
Rezk	Isomorphism-path univalence
Param'ty	
discrete	Homog. bridges express equality
. . .	. . .

A note on fibrancy

A **presheaf model** of DTT can account for:

- The **existence** of shapes (point, path, morphism, bridge, . . .)
- **Unary operations** on shapes (src, rfl)
- **Unary equations** on shapes ($\mathrm{src} \circ \mathrm{rfl} = \mathrm{id}$)

Fibrancy allows for:

- Other arities (composition, . . .)
- Specific geometries (transport, . . .)

HoTT	
Kan	Comp. of & transp. along paths
Directed	
functorial	Transport along morphisms
Segal	Composition of morphisms
Rezk	Isomorphism-path univalence
Param'ty	
discrete	Homog. bridges express equality
. . .	. . .

A note on fibrancy

A **presheaf model** of DTT can account for:

- The **existence** of shapes (point, path, morphism, bridge, . . .)
- **Unary operations** on shapes (src, rfl)
- **Unary equations** on shapes ($\mathrm{src} \circ \mathrm{rfl} = \mathrm{id}$)

Fibrancy allows for:

- Other arities (composition, . . .)
- Specific geometries (transport, . . .)

HoTT	
Kan	Comp. of & transp. along paths
Directed	
functorial	Transport along morphisms
Segal	Composition of morphisms
Rezk	Isomorphism-path univalence
Param'ty	
discrete	Homog. bridges express equality
. . .	. . .

A note on fibrancy

A **presheaf model** of DTT can account for:

- The **existence** of shapes (point, path, morphism, bridge, . . .)
- **Unary operations** on shapes (src, rfl)
- **Unary equations** on shapes ($\mathsf{src} \circ \mathsf{rfl} = \mathrm{id}$)

Fibrancy allows for:

- Other arities (composition, . . .)
- Specific geometries (transport, . . .)

HoTT	
Kan	Comp. of & transp. along paths
Directed	
functorial	Transport along morphisms
Segal	Composition of morphisms
Rezk	Isomorphism-path univalence
Param'ty	
discrete	Homog. bridges express equality
. . .	. . .

A note on fibrancy

A **presheaf model** of DTT can account for:

- The **existence** of shapes (point, path, morphism, bridge, . . .)
- **Unary operations** on shapes (src, rfl)
- **Unary equations** on shapes ($\mathrm{src} \circ \mathrm{rfl} = \mathrm{id}$)

Fibrancy allows for:

- Other arities (composition, . . .)
- Specific geometries (transport, . . .)

HoTT	
Kan	Comp. of & transp. along paths
Directed	
functorial	Transport along morphisms
Segal	Composition of morphisms
Rezk	Isomorphism-path univalence
Param'ty	
discrete	Homog. bridges express equality
. . .	. . .

A note on fibrancy

A **presheaf model** of DTT can account for:

- The **existence** of shapes (point, path, morphism, bridge, ...)
- **Unary operations** on shapes (src, rfl)
- **Unary equations** on shapes ($\mathsf{src} \circ \mathsf{rfl} = \mathrm{id}$)

Fibrancy allows for:

- Other arities (composition, ...)
- Specific geometries (transport, ...)

HoTT	
Kan	Comp. of & transp. along paths
Directed	
functorial	Transport along morphisms
Segal	Composition of morphisms
Rezk	Isomorphism-path univalence
Param'ty	
discrete	Homog. bridges express equality
...	...

A note on fibrancy

Naturality **Pre**type Theory

We **ignore** fibrancy for now:

- Functoriality & Segal fibrancy are brittle
  $\Rightarrow$ need to consider pretypes anyway
- There are promising techniques for defining fibrancy internally:
  - Contextual fibrancy [BT17, Nuy20]
  - Amazing right adjoint [LOPS18] & Transpension [ND24]
  - Internal fibrant replacement monad [Nuy20, **other?**]

HoTT	
Kan	Comp. of & transp. along paths
Directed	
functorial	Transport along morphisms
Segal	Composition of morphisms
Rezk	Isomorphism-path univalence
Param'ty	
discrete	Homog. bridges express equality
...	...

A note on fibrancy

Naturality Pretype Theory

We **ignore** fibrancy for now:

- Functoriality & Segal fibrancy are brittle
  $\Rightarrow$ need to consider pretypes anyway
- There are promising techniques for defining fibrancy internally:
  - Contextual fibrancy [BT17, Nuy20]
  - Amazing right adjoint [LOPS18] & Transpension [ND24]
  - Internal fibrant replacement monad [Nuy20, **other?**]

HoTT	
Kan	Comp. of & transp. along paths
Directed	
functorial	Transport along morphisms
Segal	Composition of morphisms
Rezk	Isomorphism-path univalence
Param'ty	
discrete	Homog. bridges express equality
...	...

A note on fibrancy

Naturality **Pre**type Theory

We **ignore** fibrancy for now:

- Functoriality & Segal fibrancy are brittle
  $\Rightarrow$ need to consider pretypes anyway
- There are promising techniques for defining fibrancy internally:
  - Contextual fibrancy [BT17, Nuy20]
  - Amazing right adjoint [LOPS18] & Transpension [ND24]
  - Internal fibrant replacement monad [Nuy20, **other?**]

HoTT	
Kan	Comp. of & transp. along paths
Directed	
functorial	Transport along morphisms
Segal	Composition of morphisms
Rezk	Isomorphism-path univalence
Param'ty	
discrete	Homog. bridges express equality
...	...

(Aside) Actually, I'd like your feedback

Definition

A CwF is **locally democratic** if every arrow $\sigma : \Delta \to \Gamma$ is isomorphic to some $\pi : \Gamma.T \to \Gamma$.

Internalizing an AWFS [§8.5 of my PhD thesis]

- A CwF is exactly a model of the **structural rules** of DTT.
- On a **locally democratic CwF**, the following **correspond**:
  - Defining an **AWFS** whose right replacement monad RR **preserves pullbacks**,
  - Modelling an **internal monad** RR on types with a **functorial action** on **dependent functions** (+ equations):

$$\frac{\Gamma, rx : \mathrm{RR}\,A \vdash T\ \mathrm{type} \qquad \Gamma \vdash f : (x : A) \to T(\eta_{\mathrm{RR}}(x))}{\Gamma \vdash \mathrm{RR}\,f : (rx : \mathrm{RR}\,A) \to (\mathrm{RR}\,T)(rx)}$$

(Aside) Actually, I'd like your feedback

Definition

A CwF is **locally democratic** if every arrow $\sigma : \Delta \to \Gamma$ is isomorphic to some $\pi : \Gamma . T \to \Gamma$.

Internalizing an AWFS [§8.5 of my PhD thesis]

- A CwF is exactly a model of the **structural rules** of DTT.
- On a **locally democratic CwF**, the following **correspond**:
  - Defining an **AWFS** whose right replacement monad RR **preserves pullbacks**,
  - Modelling an **internal monad** RR on types with a **functorial action** on **dependent functions** (+ equations):

$$\frac{\Gamma, rx : \mathrm{RR}\,A \vdash T\,\mathrm{type} \qquad \Gamma \vdash f : (x : A) \to T(\eta_{\mathrm{RR}}(x))}{\Gamma \vdash \mathrm{RR}\,f : (rx : \mathrm{RR}\,A) \to (\mathrm{RR}\,T)(rx)}$$

(Aside) Actually, I'd like your feedback

Definition

A CwF is **locally democratic** if every arrow $\sigma : \Delta \to \Gamma$ is isomorphic to some $\pi : \Gamma.T \to \Gamma$.

Internalizing an AWFS [§8.5 of my PhD thesis]

- A CwF is exactly a model of the **structural rules** of DTT.
- On a **locally democratic CwF**, the following **correspond**:
  - Defining an **AWFS** whose right replacement monad RR **preserves pullbacks**,
  - Modelling an **internal monad** RR on types
    with a **functorial action** on **dependent functions** (+ equations):

$$\frac{\begin{array}{l}\Gamma, rx : \mathrm{RR}\,A \vdash T\,\mathrm{type} \\ \Gamma \vdash f : (x : A) \to T(\eta_{\mathrm{RR}}(x))\end{array}}{\Gamma \vdash \mathrm{RR}\,f : (rx : \mathrm{RR}\,A) \to (\mathrm{RR}\,T)(rx)}$$

Model-first Approach

Separation of concerns:

We need **modalities** to keep track of **variance**.

- ➔ Instantiate **MTT** (Multimodal Type Theory) [GKNB21]
- ☺ The syntax is **their problem**!

We need **substructural intervals** for **bridges** / **morphisms** / **paths**.

- ➔ Instantiate **MTraS** (Modal Transpension System) [ND24]
- ☺ The syntax is **their problem**!
- ☺ Interaction with MTT is **their problem**!

Our concern: the **semantic requirements** for instantiating **MTT** and **MTraS**.

Model-first Approach

Separation of concerns:

We need **modalities** to keep track of **variance**.

- ➔ Instantiate **MTT** (Multimodal Type Theory) [GKNB21]
- ☺ The syntax is **their problem**!

We need **substructural intervals** for **bridges** / **morphisms** / **paths**.

- ➔ Instantiate **MTraS** (Modal Transpension System) [ND24]
- ☺ The syntax is **their problem**!
- ☺ Interaction with MTT is **their problem**!

Our concern: the **semantic requirements** for instantiating **MTT** and **MTraS**.

(Back to NatPT)

Model-first Approach

Separation of concerns:

We need **modalities** to keep track of **variance**.

- ➔ Instantiate **MTT** (Multimodal Type Theory) [GKNB21]
- ☺ The syntax is **their problem**!

We need **substructural intervals** for **bridges / morphisms / paths**.

- ➔ Instantiate **MTraS** (Modal Transpension System) [ND24]
- ☺ The syntax is **their problem**!
- ☺ Interaction with MTT is **their problem**!

Our concern: the **semantic requirements** for instantiating **MTT** and **MTraS**.

(Back to NatPT)
Model-first Approach

Separation of concerns:

We need **modalities** to keep track of **variance**.

- ➔ Instantiate **MTT** (Multimodal Type Theory) [GKNB21]
- ☺ The syntax is **their problem**!

We need **substructural intervals** for **bridges / morphisms / paths**.

- ➔ Instantiate **MTraS** (Modal Transpension System) [ND24]
- ☺ The syntax is **their problem**!
- ☺ Interaction with MTT is **their problem**!

Our concern: the **semantic requirements** for instantiating **MTT** and **MTraS**.

Towards MTT (Multimod[e/al] Type Theory)

Let $R : \mathscr{C} \to \mathscr{D}$ be a functor.

$$\frac{\Gamma \text{ ctx} @ \mathscr{C}}{R\Gamma \text{ ctx} @ \mathscr{D}} \qquad \frac{\tau : \Gamma \to \Gamma' @ \mathscr{C}}{R\tau : R\Gamma \to R\Gamma' @ \mathscr{D}} \qquad \frac{\Gamma \vdash T \text{ type} @ \mathscr{C}}{R\Gamma \vdash RT \text{ type} @ \mathscr{D}} \qquad \frac{\Gamma \vdash t : T @ \mathscr{C}}{R\Gamma \vdash Rt : RT @ \mathscr{D}}$$

Ok, so how do we check

$$\frac{?}{\Delta \vdash RT \text{ type}}$$

We check $\Gamma \vdash T \text{ type} @ \mathscr{C}$ and substitute with $\sigma : \Delta \to R\Gamma$.
BUT: Don't bother the user. Synthesize Γ and σ.

$\Gamma \in \mathscr{C}$ should be the **universal** context Γ such that $\sigma : \Delta \to R\Gamma$ exists.
I.e. if $\sigma' : \Delta \to R\Gamma'$ then we should have $\Gamma \to \Gamma'$.

+ some sensible laws $\rightsquigarrow L \dashv R$.

Let $R : \mathscr{C} \to \mathscr{D}$ be a CwF morphism.

$$\frac{\Gamma \text{ ctx} @ \mathscr{C}}{R\Gamma \text{ ctx} @ \mathscr{D}} \qquad \frac{\tau : \Gamma \to \Gamma' @ \mathscr{C}}{R\tau : R\Gamma \to R\Gamma' @ \mathscr{D}} \qquad \frac{\Gamma \vdash T \text{ type} @ \mathscr{C}}{R\Gamma \vdash RT \text{ type} @ \mathscr{D}} \qquad \frac{\Gamma \vdash t : T @ \mathscr{C}}{R\Gamma \vdash Rt : RT @ \mathscr{D}}$$

Ok, so how do we check

$$\frac{?}{\Delta \vdash RT \text{ type}}$$

We check $\Gamma \vdash T$ type $@ \mathscr{C}$ and substitute with $\sigma : \Delta \to R\Gamma$.
BUT: Don't bother the user. Synthesize Γ and σ.

$\Gamma \in \mathscr{C}$ should be the **universal** context Γ such that $\sigma : \Delta \to R\Gamma$ exists.
I.e. if $\sigma' : \Delta \to R\Gamma'$ then we should have $\Gamma \to \Gamma'$.

+ some sensible laws $\rightsquigarrow L \dashv R$.

Towards MTT (Multimod[e/al] Type Theory)

Let $R : \mathscr{C} \to \mathscr{D}$ be a CwF morphism.

$$\frac{\Gamma\ \mathsf{ctx} \mathbin{@} \mathscr{C}}{R\Gamma\ \mathsf{ctx} \mathbin{@} \mathscr{D}} \qquad \frac{\tau : \Gamma \to \Gamma' \mathbin{@} \mathscr{C}}{R\tau : R\Gamma \to R\Gamma' \mathbin{@} \mathscr{D}} \qquad \frac{\Gamma \vdash T\ \mathsf{type} \mathbin{@} \mathscr{C}}{R\Gamma \vdash RT\ \mathsf{type} \mathbin{@} \mathscr{D}} \qquad \frac{\Gamma \vdash t : T \mathbin{@} \mathscr{C}}{R\Gamma \vdash Rt : RT \mathbin{@} \mathscr{D}}$$

Ok, so how do we check

$$\frac{?}{\Delta \vdash RT\ \mathsf{type}}$$

We check $\Gamma \vdash T\ \mathsf{type} \mathbin{@} \mathscr{C}$ and substitute with $\sigma : \Delta \to R\Gamma$.
BUT: Don't bother the user. Synthesize Γ and σ.

$\Gamma \in \mathscr{C}$ should be the **universal** context Γ such that $\sigma : \Delta \to R\Gamma$ exists.
I.e. if $\sigma' : \Delta \to R\Gamma'$ then we should have $\Gamma \to \Gamma'$.

+ some sensible laws $\rightsquigarrow L \dashv R$.

Towards MTT (Multimod[e/al] Type Theory)

Let $R : \mathscr{C} \to \mathscr{D}$ be a CwF morphism.

$$\frac{\Gamma\ \mathsf{ctx} \mathbin{@} \mathscr{C}}{R\Gamma\ \mathsf{ctx} \mathbin{@} \mathscr{D}} \qquad \frac{\tau : \Gamma \to \Gamma' \mathbin{@} \mathscr{C}}{R\tau : R\Gamma \to R\Gamma' \mathbin{@} \mathscr{D}} \qquad \frac{\Gamma \vdash T\ \mathsf{type} \mathbin{@} \mathscr{C}}{R\Gamma \vdash RT\ \mathsf{type} \mathbin{@} \mathscr{D}} \qquad \frac{\Gamma \vdash t : T \mathbin{@} \mathscr{C}}{R\Gamma \vdash Rt : RT \mathbin{@} \mathscr{D}}$$

Ok, so how do we check

$$\frac{?}{\Delta \vdash RT\ \mathsf{type}}$$

We check $\Gamma \vdash T\ \mathsf{type} \mathbin{@} \mathscr{C}$ and substitute with $\sigma : \Delta \to R\Gamma$.
BUT: Don't bother the user. Synthesize Γ and σ.

$\Gamma \in \mathscr{C}$ should be the **universal** context Γ such that $\sigma : \Delta \to R\Gamma$ exists.
I.e. if $\sigma' : \Delta \to R\Gamma'$ then we should have $\Gamma \to \Gamma'$.

+ some sensible laws $\rightsquigarrow L \dashv R$.

Towards MTT (Multimod[e/al] Type Theory)

Let $R : \mathscr{C} \to \mathscr{D}$ be a CwF morphism.

$$\frac{\Gamma\ \mathrm{ctx} \,@\, \mathscr{C}}{R\Gamma\ \mathrm{ctx} \,@\, \mathscr{D}} \qquad \frac{\tau : \Gamma \to \Gamma' \,@\, \mathscr{C}}{R\tau : R\Gamma \to R\Gamma' \,@\, \mathscr{D}} \qquad \frac{\Gamma \vdash T\ \mathrm{type} \,@\, \mathscr{C}}{R\Gamma \vdash RT\ \mathrm{type} \,@\, \mathscr{D}} \qquad \frac{\Gamma \vdash t : T \,@\, \mathscr{C}}{R\Gamma \vdash Rt : RT \,@\, \mathscr{D}}$$

Ok, so how do we check

$$\frac{?}{\Delta \vdash RT\ \mathrm{type}}$$

We check $\Gamma \vdash T\ \mathrm{type} \,@\, \mathscr{C}$ and substitute with $\sigma : \Delta \to R\Gamma$.
BUT: Don't bother the user. Synthesize Γ and σ.

$\Gamma \in \mathscr{C}$ should be the **universal** context Γ such that $\sigma : \Delta \to R\Gamma$ exists.
I.e. if $\sigma' : \Delta \to R\Gamma'$ then we should have $\Gamma \to \Gamma'$.

+ some sensible laws $\rightsquigarrow L \dashv R$.

Towards MTT (Multimod[e/al] Type Theory)

Let $R : \mathscr{C} \to \mathscr{D}$ be a CwF morphism.

$$\frac{\Gamma\ \mathsf{ctx} \mathrel{@} \mathscr{C}}{R\Gamma\ \mathsf{ctx} \mathrel{@} \mathscr{D}} \qquad \frac{\tau : \Gamma \to \Gamma' \mathrel{@} \mathscr{C}}{R\tau : R\Gamma \to R\Gamma' \mathrel{@} \mathscr{D}} \qquad \frac{\Gamma \vdash T\ \mathsf{type} \mathrel{@} \mathscr{C}}{R\Gamma \vdash RT\ \mathsf{type} \mathrel{@} \mathscr{D}} \qquad \frac{\Gamma \vdash t : T \mathrel{@} \mathscr{C}}{R\Gamma \vdash Rt : RT \mathrel{@} \mathscr{D}}$$

Ok, so how do we check

$$\frac{?}{\Delta \vdash RT\ \mathsf{type}}$$

We check $\Gamma \vdash T\ \mathsf{type} \mathrel{@} \mathscr{C}$ and substitute with $\sigma : \Delta \to R\Gamma$.
BUT: Don't bother the user. Synthesize Γ and σ.

$\Gamma \in \mathscr{C}$ should be the **universal** context Γ such that $\sigma : \Delta \to R\Gamma$ exists.
I.e. if $\sigma' : \Delta \to R\Gamma'$ then we should have $\Gamma \to \Gamma'$.

+ some sensible laws $\rightsquigarrow L \dashv R$.

Let $R : \mathscr{C} \to \mathscr{D}$ be a CwF morphism.

$$\frac{\Gamma \text{ ctx} \ @\ \mathscr{C}}{R\Gamma \text{ ctx} \ @\ \mathscr{D}} \qquad \frac{\tau : \Gamma \to \Gamma' \ @\ \mathscr{C}}{R\tau : R\Gamma \to R\Gamma' \ @\ \mathscr{D}} \qquad \frac{\Gamma \vdash T \text{ type} \ @\ \mathscr{C}}{R\Gamma \vdash RT \text{ type} \ @\ \mathscr{D}} \qquad \frac{\Gamma \vdash t : T \ @\ \mathscr{C}}{R\Gamma \vdash Rt : RT \ @\ \mathscr{D}}$$

Ok, so how do we check

$$\frac{?}{\Delta \vdash RT \text{ type}}$$

We check $\Gamma \vdash T \text{ type} \ @\ \mathscr{C}$ and substitute with $\sigma : \Delta \to R\Gamma$.
BUT: Don't bother the user. Synthesize Γ and σ.

$\Gamma \in \mathscr{C}$ should be the **universal** context $L\Delta$ such that $\eta_\Delta : \Delta \to RL\Delta$ exists.
I.e. if $\sigma' : \Delta \to R\Gamma'$ then we should have $L\Delta \to \Gamma'$.

+ some sensible laws $\rightsquigarrow L \dashv R$.

Towards MTT (Multimod[e/al] Type Theory)

Let $R : \mathscr{C} \to \mathscr{D}$ be a CwF morphism.

$$\frac{\Gamma\ \mathsf{ctx} \ @\ \mathscr{C}}{R\Gamma\ \mathsf{ctx} \ @\ \mathscr{D}} \qquad \frac{\tau : \Gamma \to \Gamma' \ @\ \mathscr{C}}{R\tau : R\Gamma \to R\Gamma' \ @\ \mathscr{D}} \qquad \frac{\Gamma \vdash T\ \mathsf{type} \ @\ \mathscr{C}}{R\Gamma \vdash RT\ \mathsf{type} \ @\ \mathscr{D}} \qquad \frac{\Gamma \vdash t : T \ @\ \mathscr{C}}{R\Gamma \vdash Rt : RT \ @\ \mathscr{D}}$$

Ok, so how do we check

$$\frac{?}{\Delta \vdash RT\ \mathsf{type}}$$

We check $\Gamma \vdash T\ \mathsf{type} \ @\ \mathscr{C}$ and substitute with $\sigma : \Delta \to R\Gamma$.
BUT: Don't bother the user. Synthesize Γ and σ.

$\Gamma \in \mathscr{C}$ should be the **universal** context $L\Delta$ such that $\eta_\Delta : \Delta \to RL\Delta$ exists.
I.e. if $\sigma' : \Delta \to R\Gamma'$ then we should have $L\Delta \to \Gamma'$.

+ some sensible laws $\rightsquigarrow L \dashv R$.

MTT (Multimod[e/al] Type Theory)

MTT [GKNB21] is parametrized by a **2-category** called the **mode theory**:

- modes $p, q, r, \ldots$
- modalities $\mu : p \to q$

$$\frac{\Gamma \operatorname{ctx} @ q}{\Gamma, \text{🔒}_\mu \operatorname{ctx} @ p} \qquad \frac{\Gamma, \text{🔒}_\mu \vdash T \operatorname{type} @ p}{\Gamma \vdash \langle \mu \mid T \rangle \operatorname{type} @ q} \qquad \frac{\Gamma, \text{🔒}_\mu \vdash t : T @ p}{\Gamma \vdash \operatorname{mod}_\mu t : \langle \mu \mid T \rangle @ q}$$

- (2-cells $\alpha : \mu \Rightarrow \nu$).

Semantics:

- $[\![p]\!]$ is a (often presheaf) category modelling all of DTT,
- $[\![\mu]\!]$ is a (weak) dependent right adjoint (DRA) [BCMMPS20] to $[\![\text{🔒}_\mu]\!]$,

Note: If codomain $\mathscr{D}$ is democratic, then DRA = right adjoint that is a CwF morphism.

MTT (Multimod[e/al] Type Theory)

MTT [GKNB21] is parametrized by a **2-category** called the **mode theory**:

- modes p, q, r, ...
- modalities $\mu : p \to q$

$$\frac{\Gamma\ \mathsf{ctx} \ @\ q}{\Gamma, \text{🔒}_\mu\ \mathsf{ctx}\ @\ p} \qquad \frac{\Gamma, \text{🔒}_\mu \vdash T\ \mathsf{type}\ @\ p}{\Gamma \vdash \langle \mu \mid T \rangle\ \mathsf{type}\ @\ q} \qquad \frac{\Gamma, \text{🔒}_\mu \vdash t : T\ @\ p}{\Gamma \vdash \mathsf{mod}_\mu\, t : \langle \mu \mid T \rangle\ @\ q}$$

- (2-cells $\alpha : \mu \Rightarrow \nu$).

Semantics:

- $[\![p]\!]$ is a (often presheaf) category modelling all of DTT,
- $[\![\mu]\!]$ is a (weak) dependent right adjoint (DRA) [BCMMPS20] to $[\![\text{🔒}_\mu]\!]$,

Note: If codomain $\mathscr{D}$ is democratic, then DRA = right adjoint that is a CwF morphism.

MTT (Multimod[e/al] Type Theory)

MTT [GKNB21] is parametrized by a **2-category** called the **mode theory**:

- modes p, q, r, ...
- modalities $\mu : p \to q$

$$\frac{\Gamma \ \mathsf{ctx} \ @\ q}{\Gamma, \text{🔒}_\mu \ \mathsf{ctx} \ @\ p} \qquad \frac{\Gamma, \text{🔒}_\mu \vdash T \ \mathsf{type} \ @\ p}{\Gamma \vdash \langle \mu \mid T \rangle \ \mathsf{type} \ @\ q} \qquad \frac{\Gamma, \text{🔒}_\mu \vdash t : T \ @\ p}{\Gamma \vdash \mathsf{mod}_\mu\, t : \langle \mu \mid T \rangle \ @\ q}$$

- (2-cells $\alpha : \mu \Rightarrow \nu$).

Semantics:

- $[\![p]\!]$ is a (often presheaf) category modelling all of DTT,
- $[\![\mu]\!]$ is a (weak) dependent right adjoint (DRA) [BCMMPS20] to $[\![\text{🔒}_\mu]\!]$,

Note: If codomain $\mathscr{D}$ is democratic, then DRA = right adjoint that is a CwF morphism.

MTT (Multimod[e/al] Type Theory)

MTT [GKNB21] is parametrized by a **2-category** called the **mode theory**:

- modes p, q, r, ...
- modalities $\mu : p \to q$

$$\frac{\Gamma\,\mathsf{ctx} @ q}{\Gamma, 🔒_\mu\,\mathsf{ctx} @ p} \qquad \frac{\Gamma, 🔒_\mu \vdash T\,\mathsf{type} @ p}{\Gamma \vdash \langle \mu \mid T \rangle\,\mathsf{type} @ q} \qquad \frac{\Gamma, 🔒_\mu \vdash t : T @ p}{\Gamma \vdash \mathsf{mod}_\mu\, t : \langle \mu \mid T \rangle @ q}$$

- (2-cells $\alpha : \mu \Rightarrow \nu$).

Semantics:

- $[\![p]\!]$ is a (often presheaf) category modelling all of DTT,
- $[\![\mu]\!]$ is a (weak) dependent right adjoint (DRA) [BCMMPS20] to $[\![🔒_\mu]\!]$,

Note: If codomain $\mathscr{D}$ is democratic, then DRA = right adjoint that is a CwF morphism.

MTT (Multimod[e/al] Type Theory)

MTT [GKNB21] is parametrized by a **2-category** called the **mode theory**:

- modes p, q, r, ...
- modalities $\mu : p \to q$

$$\frac{\Gamma\ \mathsf{ctx} \ @\ q}{\Gamma, 🔒_\mu\ \mathsf{ctx} \ @\ p} \qquad \frac{\Gamma, 🔒_\mu \vdash T\ \mathsf{type} \ @\ p}{\Gamma \vdash \langle \mu \mid T \rangle\ \mathsf{type} \ @\ q} \qquad \frac{\Gamma, 🔒_\mu \vdash t : T \ @\ p}{\Gamma \vdash \mathsf{mod}_\mu\, t : \langle \mu \mid T \rangle \ @\ q}$$

- (2-cells $\alpha : \mu \Rightarrow \nu$).

Semantics:

- $[\![p]\!]$ is a (often presheaf) category modelling all of DTT,
- $[\![\mu]\!]$ is a (weak) dependent right adjoint (DRA) [BCMMPS20] to $[\![🔒_\mu]\!]$,

Note: If codomain $\mathscr{D}$ is democratic, then DRA = right adjoint that is a CwF morphism.

MTT (Multimod[e/al] Type Theory)

MTT [GKNB21] is parametrized by a **2-category** called the **mode theory**:

- modes $p, q, r, \ldots$
- modalities $\mu : p \to q$

$$\frac{\Gamma \,\mathsf{ctx} \,@\, q}{\Gamma, 🔒_\mu \,\mathsf{ctx} \,@\, p} \qquad \frac{\Gamma, 🔒_\mu \vdash T \,\mathsf{type} \,@\, p}{\Gamma \vdash \langle \mu \mid T \rangle \,\mathsf{type} \,@\, q} \qquad \frac{\Gamma, 🔒_\mu \vdash t : T \,@\, p}{\Gamma \vdash \mathsf{mod}_\mu\, t : \langle \mu \mid T \rangle \,@\, q}$$

- (2-cells $\alpha : \mu \Rightarrow \nu$).

Semantics:

- $[\![p]\!]$ is a (often presheaf) category modelling all of DTT,
- $[\![\mu]\!]$ is a (weak) dependent right adjoint (DRA) [BCMMPS20] to $[\![🔒_\mu]\!]$,

Note: If codomain $\mathscr{D}$ is democratic, then DRA = right adjoint that is a CwF morphism.

MTT (Multimod[e/al] Type Theory)

MTT [GKNB21] is parametrized by a **2-category** called the **mode theory**:

- modes $p, q, r, \ldots$
- modalities $\mu : p \to q$

$$\frac{\Gamma \,\mathsf{ctx} \,@\, q}{\Gamma, \text{🔒}_\mu \,\mathsf{ctx} \,@\, p} \qquad \frac{\Gamma, \text{🔒}_\mu \vdash T \,\mathsf{type} \,@\, p}{\Gamma \vdash \langle \mu \mid T \rangle \,\mathsf{type} \,@\, q} \qquad \frac{\Gamma, \text{🔒}_\mu \vdash t : T \,@\, p}{\Gamma \vdash \mathsf{mod}_\mu\, t : \langle \mu \mid T \rangle \,@\, q}$$

- (2-cells $\alpha : \mu \Rightarrow \nu$).

Semantics:

- $[\![p]\!]$ is a (often presheaf) category modelling all of DTT,
- $[\![\mu]\!]$ is a (weak) dependent right adjoint (DRA) [BCMMPS20] to $[\![\text{🔒}_\mu]\!]$,

Note: If codomain $\mathscr{D}$ is democratic, then DRA = right adjoint that is a CwF morphism.

Semantics of MTraS (Modal Transpension System)

Idea: Treat

$$
\begin{array}{ccccccc}
\exists(u:\mathbb{U}) & \dashv & \Finv[u] & \dashv & \forall(u:\mathbb{U}) & \dashv & \between[u] \\
\Sigma(u:\mathbb{U}) & \dashv & \Omega[u] & \dashv & \Pi(u:\mathbb{U}) & &
\end{array}
$$

as modalities.

Problem: They bind / depend on variables. (Not supported by MTT.)

Solution: Put **shape context** Ξ in the **mode**.

- $\Xi \in \mathrm{Psh}(\mathscr{W})$
- Pick **any old functor** $\sqcup \ltimes \mathbb{U} : \mathscr{W} \to \mathscr{W}$
- **Shape context extension** is $(\sqcup \ltimes \mathbb{U})_! : \mathrm{Psh}(\mathscr{W}) \to \mathrm{Psh}(\mathscr{W})$
- $\exists^{\int \Xi}_{\mathbb{U}} \dashv \Finv^{\int \Xi}_{\mathbb{U}} : \int_{\mathscr{W}} \Xi \to \int_{\mathscr{W}} (\Xi, u : \mathbb{U})$

$$
\begin{array}{ccccc}
(\exists^{\int \Xi}_{\mathbb{U}})_! & \dashv & (\exists^{\int \Xi}_{\mathbb{U}})^* & \dashv & (\exists^{\int \Xi}_{\mathbb{U}})_* \\
 & & \| \wr & & \| \wr \\
 & & (\Finv^{\int \Xi}_{\mathbb{U}})_! & \dashv & (\Finv^{\int \Xi}_{\mathbb{U}})^* & \dashv & (\Finv^{\int \Xi}_{\mathbb{U}})_*
\end{array}
$$

$$
\exists^{\Xi}_{\mathbb{U}} \dashv \Finv^{\Xi}_{\mathbb{U}} \dashv \forall^{\Xi}_{\mathbb{U}} \dashv \between^{\Xi}_{\mathbb{U}}
$$

$$
[\![\text{🔒}^{\exists u}_{\Finv u}]\!] \dashv [\![\text{🔒}^{\Finv u}_{\forall u}]\!] \dashv [\![\text{🔒}^{\forall u}_{\between u}]\!]
$$

$$
[\![\Finv u]\!] \dashv [\![\forall u]\!] \dashv [\![\between u]\!]
$$

Semantics of MTraS (Modal Transpension System)

Idea: Treat

$$
\begin{array}{ccccccc}
\exists(u : \mathbb{U}) & \dashv & \Finv[u] & \dashv & \forall(u : \mathbb{U}) & \dashv & \between[u] \\
\Sigma(u : \mathbb{U}) & \dashv & \Omega[u] & \dashv & \Pi(u : \mathbb{U}) & &
\end{array}
$$

as modalities.

Problem: They bind / depend on variables.
(Not supported by MTT.)

Solution: Put **shape context** Ξ in the **mode**.

- $\Xi \in \mathrm{Psh}(\mathscr{W})$
- Pick **any old functor** $\sqcup \ltimes \mathbb{U} : \mathscr{W} \to \mathscr{W}$
- **Shape context extension** is $(\sqcup \ltimes \mathbb{U})_! : \mathrm{Psh}(\mathscr{W}) \to \mathrm{Psh}(\mathscr{W})$
- $\exists^{\int \Xi}_{\mathbb{U}} \dashv \Finv^{\int \Xi}_{\mathbb{U}} : \int_{\mathscr{W}} \Xi \to \int_{\mathscr{W}} (\Xi, u : \mathbb{U})$

$$
\begin{array}{ccccccccc}
(\exists^{\int \Xi}_{\mathbb{U}})_! & \dashv & (\exists^{\int \Xi}_{\mathbb{U}})^* & \dashv & (\exists^{\int \Xi}_{\mathbb{U}})_* & & \\
 & & \| \wr & & \| \wr & & \\
 & & (\Finv^{\int \Xi}_{\mathbb{U}})_! & \dashv & (\Finv^{\int \Xi}_{\mathbb{U}})^* & \dashv & (\Finv^{\int \Xi}_{\mathbb{U}})_*
\end{array}
$$

$$
\exists^{\Xi}_{\mathbb{U}} \dashv \Finv^{\Xi}_{\mathbb{U}} \dashv \forall^{\Xi}_{\mathbb{U}} \dashv \between^{\Xi}_{\mathbb{U}}
$$

$$
\begin{array}{ccccccc}
[\![\text{🔒}^{\exists u}_{\Finv u}]\!] & \dashv & [\![\text{🔒}^{\Finv u}_{\forall u}]\!] & \dashv & [\![\text{🔒}^{\forall u}_{\between u}]\!] & & \\
 & & [\![\Finv u]\!] & \dashv & [\![\forall u]\!] & \dashv & [\![\between u]\!]
\end{array}
$$

Semantics of MTraS (Modal Transpension System)

Idea: Treat

$$
\begin{array}{ccccccc}
\exists(u:\mathbb{U}) & \dashv & \Finv[u] & \dashv & \forall(u:\mathbb{U}) & \dashv & \between[u] \\
\Sigma(u:\mathbb{U}) & \dashv & \Omega[u] & \dashv & \Pi(u:\mathbb{U}) & &
\end{array}
$$

as modalities.

Problem: They bind / depend on variables.
(Not supported by MTT.)

Solution: Put **shape context** Ξ in the **mode**.

- $\Xi \in \mathrm{Psh}(\mathscr{W})$
- Pick **any old functor** $\sqcup \ltimes \mathbb{U} : \mathscr{W} \to \mathscr{W}$
- **Shape context extension** is $(\sqcup \ltimes \mathbb{U})_! : \mathrm{Psh}(\mathscr{W}) \to \mathrm{Psh}(\mathscr{W})$
- $\exists^{\int \Xi}_{\mathbb{U}} \dashv \Finv^{\int \Xi}_{\mathbb{U}} : \int_{\mathscr{W}} \Xi \to \int_{\mathscr{W}} (\Xi, u : \mathbb{U})$

$$
\begin{array}{ccccccc}
(\exists^{\int \Xi}_{\mathbb{U}})_! & \dashv & (\exists^{\int \Xi}_{\mathbb{U}})^* & \dashv & (\exists^{\int \Xi}_{\mathbb{U}})_* & & \\
 & & \| \wr & & \| \wr & & \\
 & & (\Finv^{\int \Xi}_{\mathbb{U}})_! & \dashv & (\Finv^{\int \Xi}_{\mathbb{U}})^* & \dashv & (\Finv^{\int \Xi}_{\mathbb{U}})_*
\end{array}
$$

$$
\exists^{\Xi}_{\mathbb{U}} \dashv \Finv^{\Xi}_{\mathbb{U}} \dashv \forall^{\Xi}_{\mathbb{U}} \dashv \between^{\Xi}_{\mathbb{U}}
$$

$$
\begin{array}{ccccccc}
[\![\text{🔒}^{\exists u}_{\Finv u}]\!] & \dashv & [\![\text{🔒}^{\Finv u}_{\forall u}]\!] & \dashv & [\![\text{🔒}^{\forall u}_{\between u}]\!] & & \\
 & & [\![\Finv u]\!] & \dashv & [\![\forall u]\!] & \dashv & [\![\between u]\!]
\end{array}
$$

Semantics of MTraS (Modal Transpension System)

Idea: Treat

$$
\begin{array}{ccccccc}
\exists(u : \mathbb{U}) & \dashv & \reflectbox{$\exists$}[u] & \dashv & \forall(u : \mathbb{U}) & \dashv & \between[u] \\
\Sigma(u : \mathbb{U}) & \dashv & \Omega[u] & \dashv & \Pi(u : \mathbb{U}) & &
\end{array}
$$

as modalities.

Problem: They bind / depend on variables. (Not supported by MTT.)

Solution: Put **shape context** Ξ in the **mode**.

- $\Xi \in \mathrm{Psh}(\mathscr{W})$
- Pick **any old functor** $\sqcup \ltimes \mathbb{U} : \mathscr{W} \to \mathscr{W}$
- **Shape context extension** is $(\sqcup \ltimes \mathbb{U})_! : \mathrm{Psh}(\mathscr{W}) \to \mathrm{Psh}(\mathscr{W})$
- $\exists_{\mathbb{U}}^{\int \Xi} \dashv \reflectbox{$\exists$}_{\mathbb{U}}^{\int \Xi} : \int_{\mathscr{W}} \Xi \to \int_{\mathscr{W}} (\Xi, u : \mathbb{U})$

$$
\begin{array}{ccccccc}
(\exists_{\mathbb{U}}^{\int \Xi})_! & \dashv & (\exists_{\mathbb{U}}^{\int \Xi})^* & \dashv & (\exists_{\mathbb{U}}^{\int \Xi})_* & & \\
& & \| \wr & & \| \wr & & \\
& & (\reflectbox{$\exists$}_{\mathbb{U}}^{\int \Xi})_! & \dashv & (\reflectbox{$\exists$}_{\mathbb{U}}^{\int \Xi})^* & \dashv & (\reflectbox{$\exists$}_{\mathbb{U}}^{\int \Xi})_* \\
\hline\hline
\exists_{\mathbb{U}}^{\Xi} & \dashv & \reflectbox{$\exists$}_{\mathbb{U}}^{\Xi} & \dashv & \forall_{\mathbb{U}}^{\Xi} & \dashv & \between_{\mathbb{U}}^{\Xi} \\
\hline\hline
[\![\text{🔒}^{\exists u}_{\reflectbox{$\exists$} u}]\!] & \dashv & [\![\text{🔒}^{\reflectbox{$\exists$} u}_{\forall u}]\!] & \dashv & [\![\text{🔒}^{\forall u}_{\between u}]\!] & & \\
& & [\![\reflectbox{$\exists$}\, u]\!] & \dashv & [\![\forall\, u]\!] & \dashv & [\![\between\, u]\!]
\end{array}
$$

Semantics of MTraS (Modal Transpension System)

Idea: Treat

$$
\begin{array}{ccccccc}
\exists(u : \mathbb{U}) & \dashv & \Finv[u] & \dashv & \forall(u : \mathbb{U}) & \dashv & \between[u] \\
\Sigma(u : \mathbb{U}) & \dashv & \Omega[u] & \dashv & \Pi(u : \mathbb{U}) & &
\end{array}
$$

as modalities.

Problem: They bind / depend on variables. (Not supported by MTT.)

Solution: Put **shape context** Ξ in the **mode**.

- $\Xi \in \mathrm{Psh}(\mathscr{W})$
- Pick **any old functor** $\sqcup \ltimes \mathbb{U} : \mathscr{W} \to \mathscr{W}$
- **Shape context extension** is $(\sqcup \ltimes \mathbb{U})_! : \mathrm{Psh}(\mathscr{W}) \to \mathrm{Psh}(\mathscr{W})$
- $\exists^{\int \Xi}_{\mathbb{U}} \dashv \Finv^{\int \Xi}_{\mathbb{U}} : \int_{\mathscr{W}} \Xi \to \int_{\mathscr{W}} (\Xi, u : \mathbb{U})$

$$
\begin{array}{ccccccc}
(\exists^{\int \Xi}_{\mathbb{U}})_! & \dashv & (\exists^{\int \Xi}_{\mathbb{U}})^* & \dashv & (\exists^{\int \Xi}_{\mathbb{U}})_* & & \\
 & & \| \wr & & \| \wr & & \\
 & & (\Finv^{\int \Xi}_{\mathbb{U}})_! & \dashv & (\Finv^{\int \Xi}_{\mathbb{U}})^* & \dashv & (\Finv^{\int \Xi}_{\mathbb{U}})_*
\end{array}
$$

$$
\begin{array}{ccccccc}
\exists^{\Xi}_{\mathbb{U}} & \dashv & \Finv^{\Xi}_{\mathbb{U}} & \dashv & \forall^{\Xi}_{\mathbb{U}} & \dashv & \between^{\Xi}_{\mathbb{U}}
\end{array}
$$

$$
\begin{array}{ccccccc}
[\![\text{🔒}^{\exists u}_{\Finv u}]\!] & \dashv & [\![\text{🔒}^{\Finv u}_{\forall u}]\!] & \dashv & [\![\text{🔒}^{\forall u}_{\between u}]\!] & & \\
 & & [\![\Finv u]\!] & \dashv & [\![\forall u]\!] & \dashv & [\![\between u]\!]
\end{array}
$$

Semantics of MTraS (Modal Transpension System)

Idea: Treat

$$\begin{array}{ccccccc} \exists(u:\mathbb{U}) & \dashv & \text{Ǝ}[u] & \dashv & \forall(u:\mathbb{U}) & \dashv & \between[u] \\ \Sigma(u:\mathbb{U}) & \dashv & \Omega[u] & \dashv & \Pi(u:\mathbb{U}) & & \end{array}$$

as modalities.

Problem: They bind / depend on variables.
(Not supported by MTT.)

Solution: Put **shape context** Ξ in the **mode**.

- $\Xi \in \mathrm{Psh}(\mathscr{W})$
- Pick **any old functor** $\sqcup \ltimes \mathbb{U} : \mathscr{W} \to \mathscr{W}$
- **Shape context extension** is $(\sqcup \ltimes \mathbb{U})_! : \mathrm{Psh}(\mathscr{W}) \to \mathrm{Psh}(\mathscr{W})$
- $\exists^{\int \Xi}_{\mathbb{U}} \dashv \text{Ǝ}^{\int \Xi}_{\mathbb{U}} : \int_{\mathscr{W}} \Xi \to \int_{\mathscr{W}} (\Xi, u:\mathbb{U})$

$$\begin{array}{ccccccc} (\exists^{\int \Xi}_{\mathbb{U}})_! & \dashv & (\exists^{\int \Xi}_{\mathbb{U}})^* & \dashv & (\exists^{\int \Xi}_{\mathbb{U}})_* & & \\ & & \| \wr & & \| \wr & & \\ & & (\text{Ǝ}^{\int \Xi}_{\mathbb{U}})_! & \dashv & (\text{Ǝ}^{\int \Xi}_{\mathbb{U}})^* & \dashv & (\text{Ǝ}^{\int \Xi}_{\mathbb{U}})_* \end{array}$$

$$\exists^{\Xi}_{\mathbb{U}} \dashv \text{Ǝ}^{\Xi}_{\mathbb{U}} \dashv \forall^{\Xi}_{\mathbb{U}} \dashv \between^{\Xi}_{\mathbb{U}}$$

$$\begin{array}{ccccccc} [\![🔒^{\exists u}_{\text{Ǝ} u}]\!] & \dashv & [\![🔒^{\text{Ǝ} u}_{\forall u}]\!] & \dashv & [\![🔒^{\forall u}_{\between u}]\!] & & \\ & & [\![\text{Ǝ}\, u]\!] & \dashv & [\![\forall\, u]\!] & \dashv & [\![\between\, u]\!] \end{array}$$

Idea: Treat

$$
\begin{array}{ccccccc}
\exists(u:\mathbb{U}) & \dashv & \Finv[u] & \dashv & \forall(u:\mathbb{U}) & \dashv & \between[u] \\
\Sigma(u:\mathbb{U}) & \dashv & \Omega[u] & \dashv & \Pi(u:\mathbb{U}) & &
\end{array}
$$

as modalities.

Problem: They bind / depend on variables.
(Not supported by MTT.)

Solution: Put **shape context** Ξ in the **mode**.

- $\Xi \in \mathrm{Psh}(\mathscr{W})$
- Pick **any old functor** $\sqcup \ltimes \mathbb{U} : \mathscr{W} \to \mathscr{W}$
- **Shape context extension** is $(\sqcup \ltimes \mathbb{U})_! : \mathrm{Psh}(\mathscr{W}) \to \mathrm{Psh}(\mathscr{W})$
- $\exists_{\mathbb{U}}^{\int \Xi} \dashv \Finv_{\mathbb{U}}^{\int \Xi} : \int_{\mathscr{W}} \Xi \to \int_{\mathscr{W}} (\Xi, u : \mathbb{U})$

$$
\begin{array}{ccccccc}
(\exists_{\mathbb{U}}^{\int \Xi})_! & \dashv & (\exists_{\mathbb{U}}^{\int \Xi})^* & \dashv & (\exists_{\mathbb{U}}^{\int \Xi})_* & & \\
 & & \| \wr & & \| \wr & & \\
 & & (\Finv_{\mathbb{U}}^{\int \Xi})_! & \dashv & (\Finv_{\mathbb{U}}^{\int \Xi})^* & \dashv & (\Finv_{\mathbb{U}}^{\int \Xi})_* \\
\hline
\exists_{\mathbb{U}}^{\Xi} & \dashv & \Finv_{\mathbb{U}}^{\Xi} & \dashv & \forall_{\mathbb{U}}^{\Xi} & \dashv & \between_{\mathbb{U}}^{\Xi} \\
\hline
[\![🔒^{\exists u}_{\Finv u}]\!] & \dashv & [\![🔒^{\Finv u}_{\forall u}]\!] & \dashv & [\![🔒^{\forall u}_{\between u}]\!] & & \\
 & & [\![\Finv u]\!] & \dashv & [\![\forall u]\!] & \dashv & [\![\between u]\!]
\end{array}
$$

Idea: Treat

$$\begin{array}{ccccccc}
\exists(u:\mathbb{U}) & \dashv & Ⅎ[u] & \dashv & \forall(u:\mathbb{U}) & \dashv & \between[u] \\
\Sigma(u:\mathbb{U}) & \dashv & \Omega[u] & \dashv & \Pi(u:\mathbb{U}) & &
\end{array}$$

as modalities.

Problem: They bind / depend on variables.
(Not supported by MTT.)

Solution: Put **shape context** Ξ in the **mode**.

- $\Xi \in \mathrm{Psh}(\mathscr{W})$
- Pick **any old functor** $\sqcup \ltimes \mathbb{U} : \mathscr{W} \to \mathscr{W}$
- **Shape context extension** is $(\sqcup \ltimes \mathbb{U})_! : \mathrm{Psh}(\mathscr{W}) \to \mathrm{Psh}(\mathscr{W})$
- $\exists_{\mathbb{U}}^{\int \Xi} \dashv Ⅎ_{\mathbb{U}}^{\int \Xi} : \int_{\mathscr{W}} \Xi \to \int_{\mathscr{W}} (\Xi, u : \mathbb{U})$

$$\begin{array}{ccccccc}
(\exists_{\mathbb{U}}^{\int \Xi})_! & \dashv & (\exists_{\mathbb{U}}^{\int \Xi})^* & \dashv & (\exists_{\mathbb{U}}^{\int \Xi})_* & & \\
 & & \Vert \wr & & \Vert \wr & & \\
 & & (Ⅎ_{\mathbb{U}}^{\int \Xi})_! & \dashv & (Ⅎ_{\mathbb{U}}^{\int \Xi})^* & \dashv & (Ⅎ_{\mathbb{U}}^{\int \Xi})_* \\
\hline\hline
\exists_{\mathbb{U}}^{\Xi} & \dashv & Ⅎ_{\mathbb{U}}^{\Xi} & \dashv & \forall_{\mathbb{U}}^{\Xi} & \dashv & \between_{\mathbb{U}}^{\Xi} \\
\hline\hline
[\![\text{🔒}^{\exists u}_{Ⅎ u}]\!] & \dashv & [\![\text{🔒}^{Ⅎ u}_{\forall u}]\!] & \dashv & [\![\text{🔒}^{\forall u}_{\between u}]\!] & & \\
 & & [\![Ⅎ u]\!] & \dashv & [\![\forall u]\!] & \dashv & [\![\between u]\!]
\end{array}$$

Semantics of MTraS (Modal Transpension System)

Idea: Treat

$$\begin{array}{ccccccc} \exists(u:\mathbb{U}) & \dashv & \text{Ⅎ}[u] & \dashv & \forall(u:\mathbb{U}) & \dashv & \between[u] \\ \Sigma(u:\mathbb{U}) & \dashv & \Omega[u] & \dashv & \Pi(u:\mathbb{U}) & & \end{array}$$

as modalities.

Problem: They bind / depend on variables.
(Not supported by MTT.)

Solution: Put **shape context** Ξ in the **mode**.

- $\Xi \in \mathrm{Psh}(\mathscr{W})$
- Pick **any old functor** $\sqcup \ltimes \mathbb{U} : \mathscr{W} \to \mathscr{W}$
- **Shape context extension** is $(\sqcup \ltimes \mathbb{U})_! : \mathrm{Psh}(\mathscr{W}) \to \mathrm{Psh}(\mathscr{W})$
- $\exists^{\int \Xi}_{\mathbb{U}} \dashv \text{Ⅎ}^{\int \Xi}_{\mathbb{U}} : \int_{\mathscr{W}} \Xi \to \int_{\mathscr{W}} (\Xi, u : \mathbb{U})$

$$\begin{array}{ccccccccc} (\exists^{\int \Xi}_{\mathbb{U}})_! & \dashv & (\exists^{\int \Xi}_{\mathbb{U}})^* & \dashv & (\exists^{\int \Xi}_{\mathbb{U}})_* & & \\ & & \| \wr & & \| \wr & & \\ & & (\text{Ⅎ}^{\int \Xi}_{\mathbb{U}})_! & \dashv & (\text{Ⅎ}^{\int \Xi}_{\mathbb{U}})^* & \dashv & (\text{Ⅎ}^{\int \Xi}_{\mathbb{U}})_* \end{array}$$

$$\exists^{\Xi}_{\mathbb{U}} \dashv \text{Ⅎ}^{\Xi}_{\mathbb{U}} \dashv \forall^{\Xi}_{\mathbb{U}} \dashv \between^{\Xi}_{\mathbb{U}}$$

$$\begin{array}{ccccccc} [\![🔒^{\exists u}_{\text{Ⅎ} u}]\!] & \dashv & [\![🔒^{\text{Ⅎ} u}_{\forall u}]\!] & \dashv & [\![🔒^{\forall u}_{\between u}]\!] & & \\ & & [\![\text{Ⅎ} u]\!] & \dashv & [\![\forall u]\!] & \dashv & [\![\between u]\!] \end{array}$$

Introduction: Wrapping up

- We want to preserve **relations**, **morphisms** *and* **isomorphisms**.
- We need **variance** ➔ MTT
- We need **intervals** ➔ MTraS
- We need **fibrancy** ➔ future work (internal)
- For now, we care about:
  - a **mode theory**,
  - a **presheaf model** for each mode,
  - an **adjunction** for each modality.

Three Approaches to the Model

$$
\begin{tikzcd}[column sep=large, row sep=huge]
\begin{array}{c}\text{Tamsamani \& Simpson's}\\ \text{model of 2-categories}\\ \text{(2-simplicial sets)}\end{array}
  \arrow[r, "\text{heterogenize}"'] \arrow[d, "\begin{array}{c}\text{higher}\\ \text{dimension}\end{array}"]
& \begin{array}{c}\text{Pro-arrow}\\ \text{(pre-)equipments}\\ \text{[Woo82,Woo85]}\end{array}
  \arrow[d, "\begin{array}{c}\text{higher}\\ \text{dimension}\end{array}"']
& \begin{array}{c}\text{Bridge/path}\\ \text{cubical sets}\\ \text{(ParamDTT)}\\ \text{[NVD17]}\end{array}
  \arrow[l, "\text{directify}"] \arrow[d, "\begin{array}{c}\text{higher}\\ \text{dimension}\end{array}"']
\\
\begin{array}{c}\text{Tamsamani \& Simpson's}\\ \text{model of } n\text{-categories}\\ (n\text{-simplicial sets})\\ \text{[CL04,Tam99,Sim97]}\end{array}
  \arrow[r, "\text{heterogenize}"']
& \begin{array}{c}\textbf{Naturality}\\ \textbf{(Pre)type Theory}\end{array}
& \begin{array}{c}\text{Degrees of}\\ \text{Relatedness}\\ \text{(RelDTT)}\\ \text{[ND18]}\end{array}
  \arrow[l, "\text{directify}"]
\end{tikzcd}
$$

Tamsamani & Simpson's model of *n*-Categories

Tamsamani (1999)
Simpson (1997)
see Cheng & Lauda (2004)

A **reflexive graph** Γ has:

- A set of **nodes** Γ_0
- A set of **edges** Γ_1
- $\Gamma_{\mathrm{src}}, \Gamma_{\mathrm{tgt}} : \Gamma_1 \to \Gamma_0$ and $\Gamma_{\mathrm{rfl}} : \Gamma_0 \to \Gamma_1$

It is a diagram in Set:

$$\begin{tikzcd} \Gamma_0 \arrow[r, "\Gamma_{\mathrm{rfl}}"] & \Gamma_1 \arrow[l, bend right=50, "\Gamma_{\mathrm{src}}"'] \arrow[l, bend left=50, "\Gamma_{\mathrm{tgt}}"] \end{tikzcd}$$

A **simplicial set** Γ has:

- For each n, a set of ***n*-simplices** Γ_n (nodes, edges, triangles, tetrahedra, ...)
- For each monotonic $f : \{0..m\} \twoheadrightarrow \{0..n\}$, a **face map** $\Gamma_f : \Gamma_n \to \Gamma_m$ (vertices of, edges of, faces of, ...)
- For each monotonic $f : \{0..m\} \hookrightarrow \{0..n\}$, a **degeneracy map** $\Gamma_f : \Gamma_n \to \Gamma_m$ (flat tetrahedra)

It is a diagram in Set:

$$\begin{tikzcd} \Gamma_0 \arrow[r, "\Gamma_{\_\mapsto 0}"] & \Gamma_1 \arrow[l, bend right=50, "\Gamma_{0\mapsto 0}"'] \arrow[l, bend left=50, "\Gamma_{0\mapsto 1}"] \arrow[r, bend left=20] \arrow[r, bend right=20] & \Gamma_2 \arrow[l, bend right=50] \arrow[l] \arrow[l, bend left=50] & \cdots \end{tikzcd}$$

A **reflexive graph** Γ has:

- A set of **nodes** Γ_0
- A set of **edges** Γ_1
- $\Gamma_{\mathrm{src}}, \Gamma_{\mathrm{tgt}} : \Gamma_1 \to \Gamma_0$ and $\Gamma_{\mathrm{rfl}} : \Gamma_0 \to \Gamma_1$

It is a presheaf over RG:

$$\begin{tikzcd} 0 \arrow[rr, bend left, "\mathrm{src}"] \arrow[rr, bend right, "\mathrm{tgt}"'] & & 1 \arrow[ll, "\mathrm{rfl}"'] \end{tikzcd}$$

A **simplicial set** Γ has:

- For each n, a set of ***n*-simplices** Γ_n (nodes, edges, triangles, tetrahedra, ...)
- For each monotonic $f : \{0..m\} \twoheadrightarrow \{0..n\}$, a **face map** $\Gamma_f : \Gamma_n \to \Gamma_m$ (vertices of, edges of, faces of, ...)
- For each monotonic $f : \{0..m\} \hookrightarrow \{0..n\}$, a **degeneracy map** $\Gamma_f : \Gamma_n \to \Gamma_m$ (flat tetrahedra)

It is a diagram in Set:

$$\begin{tikzcd} \Gamma_0 \arrow[rr, "\Gamma_{\_\mapsto 0}"] & & \Gamma_1 \arrow[ll, bend right, "\Gamma_{0\mapsto 0}"'] \arrow[ll, bend left, "\Gamma_{0\mapsto 1}"] \arrow[rr, shift left] \arrow[rr, shift right] & & \Gamma_2 \arrow[ll, bend right] \arrow[ll] \arrow[ll, bend left] & & \cdots \end{tikzcd}$$

A **reflexive graph** Γ has:

- A set of **nodes** Γ_0
- A set of **edges** Γ_1
- $\Gamma_{\mathrm{src}}, \Gamma_{\mathrm{tgt}} : \Gamma_1 \to \Gamma_0$ and $\Gamma_{\mathrm{rfl}} : \Gamma_0 \to \Gamma_1$

A **simplicial set** Γ has:

- For each n, a set of ***n*-simplices** Γ_n
  (nodes, edges, triangles, tetrahedra, ...)
- For each monotonic
  $f : \{0..m\} \twoheadrightarrow \{0..n\}$,
  a **face map** $\Gamma_f : \Gamma_n \to \Gamma_m$
  (vertices of, edges of, faces of, ...)
- For each monotonic $f : \{0..m\} \hookrightarrow \{0..n\}$,
  a **degeneracy map** $\Gamma_f : \Gamma_n \to \Gamma_m$ (flat tetrahedra)

It is a presheaf over RG:

$$\begin{tikzcd} 0 \arrow[rr, bend left, "\mathrm{src}"] \arrow[rr, bend right, "\mathrm{tgt}"'] & & 1 \arrow[ll, "\mathrm{rfl}"'] \end{tikzcd}$$

It is a diagram in Set:

$$\begin{tikzcd}[column sep=huge] \Gamma_0 \arrow[r, "\Gamma_{\_\mapsto 0}"] & \Gamma_1 \arrow[l, bend right, "\Gamma_{0\mapsto 0}"'] \arrow[l, bend left, "\Gamma_{0\mapsto 1}"] \arrow[r, bend left=15] \arrow[r, bend right=15] & \Gamma_2 \arrow[l] \arrow[l, bend right=40] \arrow[l, bend left=40] & \cdots \end{tikzcd}$$

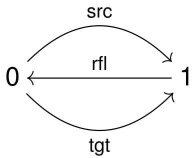
A **reflexive graph** Γ has:

- A set of **nodes** Γ_0
- A set of **edges** Γ_1
- $\Gamma_{\mathsf{src}}, \Gamma_{\mathsf{tgt}} : \Gamma_1 \to \Gamma_0$ and $\Gamma_{\mathsf{rfl}} : \Gamma_0 \to \Gamma_1$

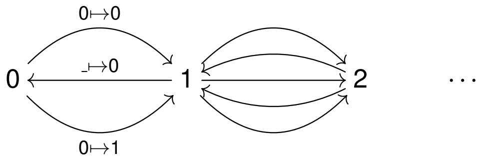
A **simplicial set** Γ has:

- For each n, a set of ***n*-simplices** Γ_n (nodes, edges, triangles, tetrahedra, ...)
- For each monotonic $f : \{0..m\} \twoheadrightarrow \{0..n\}$, a **face map** $\Gamma_f : \Gamma_n \to \Gamma_m$ (vertices of, edges of, faces of, ...)
- For each monotonic $f : \{0..m\} \hookrightarrow \{0..n\}$, a **degeneracy map** $\Gamma_f : \Gamma_n \to \Gamma_m$ (flat tetrahedra)

It is a presheaf over RG:



It is a presheaf over Δ:



Simplicial category Δ

Δ is a skeleton of NonEmptyFinLinOrd

A **reflexive graph** Γ has:

- A set of **nodes** Γ_0
- A set of **edges** Γ_1
- $\Gamma_{\mathrm{src}}, \Gamma_{\mathrm{tgt}} : \Gamma_1 \to \Gamma_0$ and $\Gamma_{\mathrm{rfl}} : \Gamma_0 \to \Gamma_1$

A **simplicial set** Γ has:

- For each n, a set of ***n*-simplices** Γ_n
  (nodes, edges, triangles, tetrahedra, …)
- For each monotonic
  $f : \{0..m\} \twoheadrightarrow \{0..n\}$,
  a **face map** $\Gamma_f : \Gamma_n \to \Gamma_m$
  (vertices of, edges of, faces of, …)
- For each monotonic $f : \{0..m\} \hookrightarrow \{0..n\}$,
  a **degeneracy map** $\Gamma_f : \Gamma_n \to \Gamma_m$ (flat tetrahedra)

It is a presheaf over RG:

$$\begin{tikzcd}
0 \arrow[r, bend left, "\mathrm{src}"] \arrow[r, bend right, "\mathrm{tgt}"'] & 1 \arrow[l, "\mathrm{rfl}"']
\end{tikzcd}$$

It is a presheaf over Δ:

$$\begin{tikzcd}
0 \arrow[r, bend left, "0\mapsto 0"] \arrow[r, bend right, "0\mapsto 1"'] & 1 \arrow[l, "_\mapsto 0"'] \arrow[r, bend left=50] \arrow[r] \arrow[r, bend right=50] & 2 \arrow[l, bend right=20] \arrow[l, bend left=20] & \cdots
\end{tikzcd}$$

Simplicial category Δ

Δ is a skeleton of NonEmptyFinLinOrd

Nerve $N(\mathscr{C})$ of a category $\mathscr{C}$

Simplicial set whose:

- **nodes** are objects
- **edges** are morphisms
- **triangles** are commutative diagrams
- **$(n \geq 3)$-simplices** uniquely exist

Segal condition

Q: When is a simplicial set the nerve of a category?

A: If every chain of n edges

$$\bullet \longrightarrow \bullet \longrightarrow \cdots \longrightarrow \bullet \longrightarrow \bullet$$

is the **spine** (Hamiltonian path) of a unique n-simplex. I.e. if compositions uniquely exist.

$$\text{Categories} \quad \simeq \quad \text{Segal simplicial sets}$$

Nerve $N(\mathscr{C})$ of a category $\mathscr{C}$

Simplicial set whose:

- **nodes** are objects
- **edges** are morphisms
- **triangles** are commutative diagrams
- $(n \geq 3)$**-simplices** uniquely exist

Segal condition

Q: When is a simplicial set the nerve of a category?

A: If every chain of n edges

$$\bullet \longrightarrow \bullet \longrightarrow \cdots \longrightarrow \bullet \longrightarrow \bullet$$

is the **spine** (Hamiltonian path) of a unique n-simplex. I.e. if compositions uniquely exist.

Categories $\simeq$ Segal simplicial sets

Nerve $N(\mathscr{C})$ of a category $\mathscr{C}$

Simplicial set whose:

- **nodes** are objects
- **edges** are morphisms
- **triangles** are commutative diagrams
- $(n \geq 3)$-**simplices** uniquely exist

Segal condition

Q: When is a simplicial set the nerve of a category?

A: If every chain of n edges

$$\bullet \longrightarrow \bullet \longrightarrow \cdots \longrightarrow \bullet \longrightarrow \bullet$$

is the **spine** (Hamiltonian path) of a unique n-simplex. I.e. if compositions uniquely exist.

$$\text{Categories} \quad \simeq \quad \text{Segal simplicial sets}$$

Let $(\mathscr{V}, I, \otimes)$ be a **monoidal category**.

A $\mathscr{V}$**-enriched** category $\mathscr{C}$ has:

- A (big) set of **objects**
- For each $x, y \in \mathrm{Obj}(\mathscr{C})$, a **Hom-thing** $\mathrm{Hom}(x, y) \in \mathrm{Obj}(\mathscr{V})$,
- $\mathrm{id}_x : I \to \mathrm{Hom}(x, x)$
- $\circ : \mathrm{Hom}(y, z) \otimes \mathrm{Hom}(x, y) \to \mathrm{Hom}(x, z)$

Strict n-category

- A 0-category is a **set**.
- An $(n+1)$-category is a **category enriched over n-categories**.

Q: Can we understand higher categories via simplicial sets?

Cheng & Lauda's Guidebook: [CL04]
A thousand times **yes!**

Tamsamani & Simpson: [Sim97,Tam99]
One such time **yes!**
➔ using **double** / n**-fold** categories

Let $(\mathscr{V}, \top, \times)$ be a **cartesian category**.

A $\mathscr{V}$-enriched category $\mathscr{C}$ has:

- A (big) set of **objects**
- For each $x, y \in \mathrm{Obj}(\mathscr{C})$, a **Hom-thing** $\mathrm{Hom}(x, y) \in \mathrm{Obj}(\mathscr{V})$,
- $\mathrm{id}_x : \top \to \mathrm{Hom}(x, x)$
- $\circ : \mathrm{Hom}(y, z) \times \mathrm{Hom}(x, y) \to \mathrm{Hom}(x, z)$

Strict n-category

- A 0-category is a **set**.
- An $(n+1)$-category is a **category enriched over n-categories**.

Q: Can we understand higher categories via simplicial sets?

Cheng & Lauda's Guidebook: [CL04]
A thousand times **yes!**

Tamsamani & Simpson: [Sim97,Tam99]
One such time **yes!**
➔ using **double** / ***n*-fold** categories

Let $(\mathscr{V}, \top, \times)$ be a **cartesian category**.

A $\mathscr{V}$-enriched category $\mathscr{C}$ has:

- A (big) set of **objects**
- For each $x, y \in \mathrm{Obj}(\mathscr{C})$, a **Hom-thing** $\mathrm{Hom}(x,y) \in \mathrm{Obj}(\mathscr{V})$,
- $\mathrm{id}_x : \top \to \mathrm{Hom}(x,x)$
- $\circ : \mathrm{Hom}(y,z) \times \mathrm{Hom}(x,y) \to \mathrm{Hom}(x,z)$

Strict n-category

- A 0-category is a **set**.
- An $(n+1)$-category is a **category enriched over n-categories**.

Q: Can we understand higher categories via simplicial sets?

Cheng & Lauda's Guidebook: [CL04]
A thousand times **yes!**

Tamsamani & Simpson: [Sim97,Tam99]
One such time **yes!**
➔ using **double** / **n-fold** categories

Let $(\mathscr{V}, \top, \times)$ be a **cartesian category**.

A $\mathscr{V}$-**enriched** category $\mathscr{C}$ has:

- A (big) set of **objects**
- For each $x, y \in \mathrm{Obj}(\mathscr{C})$, a **Hom-thing** $\mathrm{Hom}(x,y) \in \mathrm{Obj}(\mathscr{V})$,
- $\mathrm{id}_x : \top \to \mathrm{Hom}(x,x)$
- $\circ : \mathrm{Hom}(y,z) \times \mathrm{Hom}(x,y) \to \mathrm{Hom}(x,z)$

Strict *n*-category

- A 0-category is a **set**.
- An $(n+1)$-category is a **category enriched over *n*-categories**.

Q: Can we understand higher categories via simplicial sets?

Cheng & Lauda's Guidebook: [CL04]
A thousand times **yes!**

Tamsamani & Simpson: [Sim97,Tam99]
One such time **yes!**
➔ using **double** / ***n*-fold** categories

A **double category** $\mathscr{C}$ has:

- objects
- horiz. arrows / (1)-arrows (1-cells)
- vertical arrows / (2)-arrows (trivial)
- squares (2-cells)

and can be defined as a **bisimplicial set** $\mathscr{C} \in \mathrm{Psh}(\Delta \times \Delta)$ satisfying the **Segal condition** in each dimension.

$$
\bullet \qquad \begin{tikzcd} \bullet \arrow[r] & \bullet \end{tikzcd}
$$

$$
\begin{tikzcd} \bullet \arrow[d] \\ \bullet \end{tikzcd} \qquad \begin{tikzcd} \bullet \arrow[r] \arrow[d] & \bullet \arrow[d] \\ \bullet \arrow[r] & \bullet \end{tikzcd}
$$

A T&S 2-category is:

a double category whose vertical arrows are trivial.

A **double category** $\mathscr{C}$ has:

- objects
- horiz. arrows / (1)-arrows (1-cells)
- vertical arrows / (2)-arrows (trivial)
- squares (2-cells)

and can be defined as a **bisimplicial set** $\mathscr{C} \in \mathrm{Psh}(\Delta \times \Delta)$ satisfying the **Segal condition** in each dimension.

A **T&S 2-category** is:

a **double category** whose **vertical arrows** are **trivial**.

$$\begin{tikzcd} \bullet & \bullet \arrow[r] & \bullet \end{tikzcd}$$

$$\begin{tikzcd} \bullet \arrow[d] & \bullet \arrow[r] \arrow[d] & \bullet \arrow[d] \\ \bullet & \bullet \arrow[r] & \bullet \end{tikzcd}$$

A **double category** $\mathscr{C}$ has:

- objects
- horiz. arrows / (1)-arrows (1-cells)
- vertical arrows / (2)-arrows (trivial)
- squares (2-cells)

and can be defined as a **bisimplicial set** $\mathscr{C} \in \mathrm{Psh}(\Delta \times \Delta)$ satisfying the **Segal condition** in each dimension.

A **T&S 2-category** is:

a **double category** whose **vertical arrows** are **trivial**.

$$\begin{tikzcd} \bullet & \bullet \arrow[r] & \bullet \end{tikzcd}$$

$$\begin{tikzcd} \bullet \arrow[d] & \bullet \arrow[r] \arrow[d] & \bullet \arrow[d] \\ \bullet & \bullet \arrow[r] & \bullet \end{tikzcd}$$

A **double category** $\mathscr{C}$ has:

- objects
- horiz. arrows / (1)-arrows (1-cells)
- vertical arrows / (2)-arrows (trivial)
- squares (2-cells)

and can be defined as a **bisimplicial set** $\mathscr{C} \in \mathrm{Psh}(\Delta \times \Delta)$ satisfying the **Segal condition** in each dimension.

$$\bullet \qquad \bullet \longrightarrow \bullet$$

$$\begin{tikzcd} \bullet \arrow[d, equal] \\ \bullet \end{tikzcd} \qquad \begin{tikzcd} \bullet \arrow[r] \arrow[d, equal] & \bullet \arrow[d, equal] \\ \bullet \arrow[r] & \bullet \end{tikzcd}$$

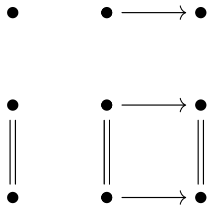
A **T&S 2-category** is:

a **double category** whose **vertical arrows** are **trivial**.

A **double category** $\mathscr{C}$ has:

- objects
- horiz. arrows / (1)-arrows (1-cells)
- vertical arrows / (2)-arrows (trivial)
- squares (2-cells)

and can be defined as a **bisimplicial set** $\mathscr{C} \in \mathrm{Psh}(\Delta \times \Delta)$ satisfying the **Segal condition** in each dimension.



A **T&S 2-category** is:

a **double category** whose **vertical arrows** are **trivial**.

A double category $\mathscr{C}$ has:

- objects
- horiz. arrows / (1)-arrows (1-cells)
- vertical arrows / (2)-arrows (trivial)
- squares (2-cells)

and can be defined as a **bisimplicial set** $\mathscr{C} \in \mathrm{Psh}(\Delta \times \Delta)$ satisfying the **Segal condition** in each dimension.

A T&S 2-category is:

a **double category** whose **vertical arrows** are **trivial**.

An *n*-fold category is:

an ***n*-simplicial set** $\mathscr{C} \in \mathrm{Psh}(\Delta^n)$ satisfying the **Segal condition** in each dimension.

A T&S *n*-category is:

an ***n*-fold** category where only

- (1)-arrows, (1-cells)
- (1,2)-squares, (2-cells)
- (1,2,3)-cubes, (3-cells)
- ...

can be non-trivial.

Pretypes!

A **double category** $\mathscr{C}$ has:

- objects
- horiz. arrows / (1)-arrows (1-cells)
- vertical arrows / (2)-arrows (trivial)
- squares (2-cells)

and can be defined as a **bisimplicial set** $\mathscr{C} \in \mathrm{Psh}(\Delta \times \Delta)$ satisfying the **Segal condition** in each dimension.

A **T&S 2-category** is:

a **double category** whose **vertical arrows** are **trivial**.

An ***n*-fold category** is:

an ***n*-simplicial set** $\mathscr{C} \in \mathrm{Psh}(\Delta^n)$ satisfying the **Segal condition** in each dimension.

A **T&S *n*-category** is:

an ***n*-fold** category where only

- (1)-arrows, (1-cells)
- (1,2)-squares, (2-cells)
- (1,2,3)-cubes, (3-cells)
- $\ldots$

can be non-trivial.

Pretypes!

A **double category** $\mathscr{C}$ has:

- objects
- horiz. arrows / (1)-arrows (1-cells)
- vertical arrows / (2)-arrows (trivial)
- squares (2-cells)

and can be defined as a **bisimplicial set** $\mathscr{C} \in \mathrm{Psh}(\Delta \times \Delta)$ satisfying the **Segal condition** in each dimension.

A **T&S 2-category** is:

a **double category** whose **vertical arrows** are **trivial**.

An *n*-**fold category** is:

an *n*-**simplicial set** $\mathscr{C} \in \mathrm{Psh}(\Delta^n)$ satisfying the **Segal condition** in each dimension.

A **T&S** *n*-**category** is:

an *n*-**fold** category where only

- (1)-arrows, (1-cells)
- (1,2)-squares, (2-cells)
- (1,2,3)-cubes, (3-cells)
- $\ldots$

can be non-trivial.

Pretypes!

Three Approaches to the Model

$$
\begin{tikzcd}[column sep=large, row sep=huge]
\begin{array}{c}\text{Tamsamani \& Simpson's}\\ \text{model of 2-categories}\\ \text{(2-simplicial sets)}\end{array} \arrow[r, "\text{heterogenize}"'] \arrow[d, "\begin{array}{c}\text{higher}\\ \text{dimension}\end{array}"] & \begin{array}{c}\text{Pro-arrow}\\ \text{(pre-)equipments}\\ \text{[Woo82,Woo85]}\end{array} \arrow[d, "\begin{array}{c}\text{higher}\\ \text{dimension}\end{array}"'] & \begin{array}{c}\text{Bridge/path}\\ \text{cubical sets}\\ \text{(ParamDTT)}\\ \text{[NVD17]}\end{array} \arrow[l, "\text{directify}"] \arrow[d, "\begin{array}{c}\text{higher}\\ \text{dimension}\end{array}"'] \\
\begin{array}{c}\text{Tamsamani \& Simpson's}\\ \text{model of } n\text{-categories}\\ (n\text{-simplicial sets})\\ \text{[CL04,Tam99,Sim97]}\end{array} \arrow[r, "\text{heterogenize}"'] & \begin{array}{c}\textbf{Naturality}\\ \textbf{(Pre)type Theory}\end{array} & \begin{array}{c}\text{Degrees of}\\ \text{Relatedness}\\ \text{(RelDTT)}\\ \text{[ND18]}\end{array} \arrow[l, "\text{directify}"]
\end{tikzcd}
$$

Pro-arrow
Equipments

Richard J. Wood (1982, 1985)

(Pro-arrow) Equipment

An **equipment** $\mathscr{C}$ is a **double category** with

- objects
- arrows ($\to$)
- pro-arrows ($\not\to$)
- squares

such that every arrow $\varphi : x \to y$ has **"graph"** pro-arrows

$$\varphi^{\ddagger} : x \not\to y, \qquad \varphi^{\dagger} : y \not\to x$$

such that $(\ldots)$.

Example (Set)

Set is an **equipment** with:

- sets
- functions
- relations
  - identity relation: equality
  - $(R; S)(x, z) = \exists y . R(x, y) \wedge S(y, z)$
- proofs that $R(a, b) \Rightarrow S(f\,a, g\,b)$

$$\begin{tikzcd} A \arrow[r, "R", "\shortmid"{marking}] \arrow[d, "f"'] & B \arrow[d, "g"] \\ C \arrow[r, "S"', "\shortmid"{marking}] & D \end{tikzcd}$$

(Pro-arrow) Equipment

An **equipment** $\mathscr{C}$ is a **double category** with

- objects
- arrows ($\to$)
- pro-arrows ($\not\to$)
- squares

such that every arrow $\varphi : x \to y$ has **“graph”** pro-arrows

$$\varphi^{\ddagger} : x \not\to y, \qquad \varphi^{\dagger} : y \not\to x$$

such that $(\ldots)$.

Example (Set)

Set is an **equipment** with:

- sets
- functions
- relations
  - identity relation: equality
  - $(R; S)(x, z) = \exists y . R(x, y) \wedge S(y, z)$
- proofs that $R(a, b) \Rightarrow S(f\,a, g\,b)$

$$\begin{tikzcd} A \arrow[r, "R", "\shortmid"{marking}] \arrow[d, "f"'] & B \arrow[d, "g"] \\ C \arrow[r, "S"', "\shortmid"{marking}] & D \end{tikzcd}$$

(Pro-arrow) Equipment

An **equipment** $\mathscr{C}$ is a **double category** with

- objects
- arrows ($\rightarrow$)
- pro-arrows ($\nrightarrow$)
- squares

such that every arrow $\varphi : x \rightarrow y$ has **"graph"** pro-arrows

$$\varphi^{\ddagger} : x \nrightarrow y, \qquad \varphi^{\dagger} : y \nrightarrow x$$

such that $(\ldots)$.

Example (Set)

Set is an **equipment** with:

- sets
- functions
- relations
  - identity relation: equality
  - $(R; S)(x, z) = \exists y . R(x, y) \wedge S(y, z)$
- proofs that $R(a, b) \Rightarrow S(f\, a, g\, b)$

$$\begin{tikzcd} A \arrow[r, "R", "\shortmid"{marking}] \arrow[d, "f"'] & B \arrow[d, "g"] \\ C \arrow[r, "S"', "\shortmid"{marking}] & D \end{tikzcd}$$

(Pro-arrow) Equipment

An **equipment** $\mathscr{C}$ is a **double category** with

- objects
- arrows ($\to$)
- pro-arrows ($\not\to$)
- squares

such that every arrow $\varphi : x \to y$ has **“graph”** pro-arrows

$$\varphi^{\ddagger} : x \not\to y, \qquad \varphi^{\dagger} : y \not\to x$$

such that $(\ldots)$.

Example (Cat)

Cat is an **equipment** with:

- categories
- functors
- profunctors $\mathscr{P} : \mathscr{A}^{\mathrm{op}} \times \mathscr{B} \to \mathrm{Set}$
  - identity profunctor: Hom
  - $(\mathscr{P};\mathscr{Q})(x,z) = \overset{\text{coend}}{\exists}\, y.\mathscr{P}(x,y) \times \mathscr{Q}(y,z)$
- $\overset{\text{end}}{\forall}\, a,b.\mathscr{P}(a,b) \Rightarrow \mathscr{Q}(F\,a, G\,b)$

$$\begin{tikzcd}
\mathscr{A} \arrow[r, "\mathscr{P}", "|" marking] \arrow[d, "F"'] & \mathscr{B} \arrow[d, "G"] \\
\mathscr{C} \arrow[r, "\mathscr{Q}"', "|" marking] & \mathscr{D}
\end{tikzcd}$$

Example (Cat)

Cat is a **T&S 2-category** with:

- categories
- functors
- trivial (2)-arrows
- nat. transformations $\overset{\text{end}}{\forall}\, a.\mathrm{Hom}(F\,a, G\,a)$

i.e. $\overset{\text{end}}{\forall}\, a, b.\mathrm{Hom}(a, b) \Rightarrow \mathrm{Hom}(F\,a, G\,b)$

$$\begin{tikzcd} \mathscr{A} \arrow[r, "\mathrm{Hom}", "|"] \arrow[d, "F"'] & \mathscr{A} \arrow[d, "G"] \\ \mathscr{C} \arrow[r, "\mathrm{Hom}"', "|"] & \mathscr{C} \end{tikzcd}$$

To get **heterogeneous** nat. transformations: **drop** T&S's triviality condition!

Example (Cat)

Cat is an **equipment** with:

- categories
- functors
- profunctors $\mathscr{P} : \mathscr{A}^{\text{op}} \times \mathscr{B} \to \text{Set}$
  - identity profunctor: Hom
  - $(\mathscr{P}; \mathscr{Q})(x, z) = \overset{\text{coend}}{\exists}\, y.\mathscr{P}(x, y) \times \mathscr{Q}(y, z)$
- $\overset{\text{end}}{\forall}\, a, b.\mathscr{P}(a, b) \Rightarrow \mathscr{Q}(F\,a, G\,b)$

$$\begin{tikzcd} \mathscr{A} \arrow[r, "\mathscr{P}", "|"] \arrow[d, "F"'] & \mathscr{B} \arrow[d, "G"] \\ \mathscr{C} \arrow[r, "\mathscr{Q}"', "|"] & \mathscr{D} \end{tikzcd}$$

Example (Cat)

Cat is a **T&S 2-category** with:

- categories
- functors
- trivial (2)-arrows
- nat. transformations $\overset{\text{end}}{\forall}\, a.\mathrm{Hom}(F\,a, G\,a)$
  i.e. $\overset{\text{end}}{\forall}\, a, b.\mathrm{Hom}(a, b) \Rightarrow \mathrm{Hom}(F\,a, G\,b)$

$$\begin{tikzcd}
\mathscr{A} \arrow[r, "\mathrm{Hom}", "\shortmid"{marking}] \arrow[d, "F"'] & \mathscr{A} \arrow[d, "G"] \\
\mathscr{C} \arrow[r, "\mathrm{Hom}"', "\shortmid"{marking}] & \mathscr{C}
\end{tikzcd}$$

To get **heterogeneous** nat. transformations:
drop T&S's triviality condition!

Example (Cat)

Cat is an **equipment** with:

- categories
- functors
- profunctors $\mathscr{P} : \mathscr{A}^{\mathrm{op}} \times \mathscr{B} \to \mathsf{Set}$
  - identity profunctor: Hom
  - $(\mathscr{P}; \mathscr{Q})(x, z) = \overset{\text{coend}}{\exists}\, y.\mathscr{P}(x, y) \times \mathscr{Q}(y, z)$
- $\overset{\text{end}}{\forall}\, a, b.\mathscr{P}(a, b) \Rightarrow \mathscr{Q}(F\,a, G\,b)$

$$\begin{tikzcd}
\mathscr{A} \arrow[r, "\mathscr{P}", "\shortmid"{marking}] \arrow[d, "F"'] & \mathscr{B} \arrow[d, "G"] \\
\mathscr{C} \arrow[r, "\mathscr{Q}"', "\shortmid"{marking}] & \mathscr{D}
\end{tikzcd}$$

Example (Cat)

Cat is a **T&S 2-category** with:

- categories
- functors
- trivial (2)-arrows
- nat. transformations $\overset{\text{end}}{\forall}\, a.\mathrm{Hom}(F\,a, G\,a)$

  i.e. $\overset{\text{end}}{\forall}\, a, b.\mathrm{Hom}(a, b) \Rightarrow \mathrm{Hom}(F\,a, G\,b)$

$$\begin{tikzcd}
\mathscr{A} \arrow[r, "\mathrm{Hom}", "|" marking] \arrow[d, "F"'] & \mathscr{A} \arrow[d, "G"] \\
\mathscr{C} \arrow[r, "\mathrm{Hom}"', "|" marking] & \mathscr{C}
\end{tikzcd}$$

To get **heterogeneous** nat. transformations:
drop T&S's triviality condition!

Example (Cat)

Cat is an **equipment** with:

- categories
- functors
- profunctors $\mathscr{P} : \mathscr{A}^{\mathsf{op}} \times \mathscr{B} \to \mathsf{Set}$
  - identity profunctor: Hom
  - $(\mathscr{P}; \mathscr{Q})(x, z) =$

    $\overset{\text{coend}}{\exists}\, y.\mathscr{P}(x, y) \times \mathscr{Q}(y, z)$
- $\overset{\text{end}}{\forall}\, a, b.\mathscr{P}(a, b) \Rightarrow \mathscr{Q}(F\,a, G\,b)$

$$\begin{tikzcd}
\mathscr{A} \arrow[r, "\mathscr{P}", "|" marking] \arrow[d, "F"'] & \mathscr{B} \arrow[d, "G"] \\
\mathscr{C} \arrow[r, "\mathscr{Q}"', "|" marking] & \mathscr{D}
\end{tikzcd}$$

Three Approaches to the Model

$$\begin{tikzcd}[column sep=huge, row sep=huge]
\begin{array}{c}\text{Tamsamani \& Simpson's}\\ \text{model of 2-categories}\\ \text{(2-simplicial sets)}\end{array} \arrow[r, "\text{heterogenize}"'] \arrow[d, "\begin{array}{c}\text{higher}\\ \text{dimension}\end{array}"] & \begin{array}{c}\text{Pro-arrow}\\ \text{(pre-)equipments}\\ \text{[Woo82,Woo85]}\end{array} \arrow[d, "\begin{array}{c}\text{higher}\\ \text{dimension}\end{array}"'] & \begin{array}{c}\text{Bridge/path}\\ \text{cubical sets}\\ \text{(ParamDTT)}\\ \text{[NVD17]}\end{array} \arrow[l, "\text{directify}"] \arrow[d, "\begin{array}{c}\text{higher}\\ \text{dimension}\end{array}"'] \\
\begin{array}{c}\text{Tamsamani \& Simpson's}\\ \text{model of } n\text{-categories}\\ (n\text{-simplicial sets})\\ \text{[CL04,Tam99,Sim97]}\end{array} \arrow[r, "\text{heterogenize}"'] & \begin{array}{c}\textbf{Naturality}\\ \textbf{(Pre)type Theory}\end{array} & \begin{array}{c}\text{Degrees of}\\ \text{Relatedness}\\ \text{(RelDTT)}\\ \text{[ND18]}\end{array} \arrow[l, "\text{directify}"]
\end{tikzcd}$$

Higher
Pro-arrow
Equipments

Higher Pro-arrow Equipments

Set is . . .

- ☹ A large set
- 😐 A category
- ☺ An equipment

Cat is . . .

- ☹ A category
- 😐 A 2-category
- ☺ An equipment

Eqmnt is . . .

- ☹ An equipment
- ☺ A 2-equipment

Eqmnt has:

Objects	Equipments
Arrows	Equipment functors
Pro-arrows	Equipment profunctors: Contain arrows and pro-arrows
Pro-pro-arrows	Equipment **pro-profunctors:** Contain pro-arrows
Squares	. . .
Cubes	. . .

Higher Equipment

An n-**equipment** is an n-**fold category** (. . .)

$\Rightarrow \mathscr{C} \in \mathrm{Psh}(\Delta^n_{\dagger,\ddagger})$

Higher Pro-arrow Equipments

Set is . . .

- ☹ A large set
- 😐 A category
- ☺ An equipment

Cat is . . .

- ☹ A category
- 😐 A 2-category
- ☺ An equipment

Eqmnt is . . .

- ☹ An equipment
- ☺ A 2-equipment

Eqmnt has:

Objects	Equipments
Arrows	Equipment functors
Pro-arrows	Equipment profunctors: Contain arrows and pro-arrows
Pro-pro-arrows	Equipment **pro-profunctors:** Contain pro-arrows
Squares	. . .
Cubes	. . .

Higher Equipment

An n**-equipment** is an n**-fold category** (. . .)

$\Rightarrow \mathscr{C} \in \mathrm{Psh}(\Delta^n_{\dagger,\ddagger})$

Higher Pro-arrow Equipments

Set is . . .

- ☹ A large set
- 😐 A category
- ☺ An equipment

Cat is . . .

- ☹ A category
- 😐 A 2-category
- ☺ An equipment

Eqmnt is . . .

- ☹ An equipment
- ☺ A 2-equipment

Eqmnt has:

Objects	Equipments
Arrows	Equipment functors
Pro-arrows	Equipment profunctors: Contain arrows and pro-arrows
Pro-pro-arrows	Equipment **pro-profunctors:** Contain pro-arrows
Squares	. . .
Cubes	. . .

Higher Equipment

An n-**equipment** is an n-**fold category** (. . .)

$\Rightarrow \mathscr{C} \in \mathrm{Psh}(\Delta^n_{\dagger,\ddagger})$

Higher Pro-arrow Equipments

Set is . . .

- ☹ A large set
- 😐 A category
- ☺ An equipment

Cat is . . .

- ☹ A category
- 😐 A 2-category
- ☺ An equipment

Eqmnt is . . .

- ☹ An equipment
- ☺ A 2-equipment

Eqmnt has:

Objects	Equipments
Arrows	Equipment functors
Pro-arrows	Equipment profunctors: Contain arrows and pro-arrows
Pro-pro-arrows	Equipment **pro-profunctors:** Contain pro-arrows
Squares	. . .
Cubes	. . .

Higher Equipment

An n-**equipment** is an n-**fold category** (. . .)

$\Rightarrow \mathscr{C} \in \mathrm{Psh}(\Delta^n_{\dagger,\ddagger})$

Higher Pro-arrow Equipments

Set is . . .

- ☹ A large set
- 😐 A category
- ☺ An equipment

Cat is . . .

- ☹ A category
- 😐 A 2-category
- ☺ An equipment

Eqmnt is . . .

- ☹ An equipment
- ☺ A 2-equipment

Eqmnt has:

Objects	Equipments
Arrows	Equipment functors
Pro-arrows	Equipment profunctors: Contain arrows and pro-arrows
Pro-pro-arrows	Equipment **pro-profunctors:** Contain pro-arrows
Squares	. . .
Cubes	. . .

Higher Equipment

An n**-equipment** is an n**-fold category** (. . .)

$$\Rightarrow \mathscr{C} \in \mathrm{Psh}(\Delta^n_{\dagger,\ddagger})$$

“Equipment pro-profunctors”!?
Are you making this up?

✓ Only partially.

Hofmann-Streicher Universe [HS97]

For any category $\mathscr{W}$,
$\mathrm{Psh}(\mathscr{W})$ models **DTT, with a universe** U^{HS}.

Let $W \in \mathrm{Obj}(\mathscr{W})$.
A W-cell of U^{HS} contains:

- a notion of dependent W-cells

Looking at this differently

Define $\dot{\mathscr{W}} :\cong \mathscr{W}$.
If $W \in \mathrm{Obj}(\mathscr{W})$, then pro $W \in \mathrm{Obj}(\dot{\mathscr{W}})$.

Consider $\mathsf{U}^{\mathsf{HS}}_{\dot{\mathscr{W}}} \in \mathrm{Psh}(\dot{\mathscr{W}})$.

pro W-cells have a different meaning:

- **Param’ty:** U of discrete types is not discrete.
  ➔ Edges express het. equality;
  pro-edges express **relations**.
- **Directed:** U of Segal types is not Segal.
  ➔ Arrows express morphisms;
  pro-arrows express **profunctors**.
  ➔ Triangles express commutativity;
  pro-triangles are **boundary predicates**.

"Equipment pro-profunctors"!? Are you making this up?

Only partially.

Hofmann-Streicher Universe [HS97]

For any category $\mathscr{W}$,
$\mathrm{Psh}(\mathscr{W})$ models **DTT, with a universe** U^{HS}.

Let $W \in \mathrm{Obj}(\mathscr{W})$.
A pro W**-cell** of U^{HS} contains:

- a notion of dependent W-cells
- for all $V \in \mathrm{Obj}(\mathscr{W}/W)$, a notion of dependent V-cells

Looking at this differently

Define $\dot{\mathscr{W}} :\cong \mathscr{W}$.
If $W \in \mathrm{Obj}(\mathscr{W})$, then pro $W \in \mathrm{Obj}(\dot{\mathscr{W}})$.

Consider $\mathrm{U}^{\mathrm{HS}}_{\mathscr{W}} \in \mathrm{Psh}(\dot{\mathscr{W}})$.

pro W-cells have a different meaning:

- **Param'ty:** U of discrete types is not discrete.
  Edges express het. equality;
  pro-edges express **relations**.
- **Directed:** U of Segal types is not Segal.
  Arrows express morphisms;
  pro-arrows express **profunctors**.
  Triangles express commutativity;
  pro-triangles are **boundary predicates**.

“Equipment pro-profunctors”!? Are you making this up?

Only partially.

Hofmann-Streicher Universe [HS97]

For any category $\mathscr{W}$,
$\mathrm{Psh}(\mathscr{W})$ models **DTT, with a universe U^{HS}.**

Let $W \in \mathrm{Obj}(\mathscr{W})$.
A pro W**-cell** of U^{HS} contains:

- a notion of dependent W-cells
- for all $V \in \mathrm{Obj}(\mathscr{W}/W)$, a notion of dependent V-cells

Looking at this differently

Define $\dot{\mathscr{W}} :\cong \mathscr{W}$.
If $W \in \mathrm{Obj}(\mathscr{W})$, then pro $W \in \mathrm{Obj}(\dot{\mathscr{W}})$.

Consider $\mathsf{U}^{\mathsf{HS}}_{\mathscr{W}} \in \mathrm{Psh}(\dot{\mathscr{W}})$.

pro W-cells have a different meaning:

- **Param'ty:** U of discrete types is not discrete.
  ➔ Edges express het. equality; **pro**-edges express **relations**.
- **Directed:** U of Segal types is not Segal.
  ➔ Arrows express morphisms; pro-arrows express profunctors.
  ➔ Triangles express commutativity; pro-triangles are boundary predicates.

"Equipment pro-profunctors"!?
Are you making this up?

Only partially.

Hofmann-Streicher Universe [HS97]

For any category $\mathscr{W}$,
$\mathrm{Psh}(\mathscr{W})$ models **DTT, with a universe U^{HS}.**

Let $W \in \mathrm{Obj}(\mathscr{W})$.
A pro W**-cell** of U^{HS} contains:

- a notion of dependent W-cells
- for all $V \in \mathrm{Obj}(\mathscr{W}/W)$, a notion of dependent V-cells

Looking at this differently

Define $\dot{\mathscr{W}} :\cong \mathscr{W}$.
If $W \in \mathrm{Obj}(\mathscr{W})$, then pro $W \in \mathrm{Obj}(\dot{\mathscr{W}})$.

Consider $\mathrm{U}^{\mathrm{HS}}_{\mathscr{W}} \in \mathrm{Psh}(\dot{\mathscr{W}})$.

pro W-cells have a different meaning:

- **Param'ty:** U of discrete types is not discrete.
  - Edges express het. equality; pro-edges express **relations**.
- **Directed:** U of Segal types is not Segal.
  - Arrows express morphisms; pro-arrows express **profunctors**.
  - Triangles express commutativity; pro-triangles are **boundary predicates**.

“Equipment pro-profunctors”!? Are you making this up?

Only partially.

Hofmann-Streicher Universe [HS97]

For any category $\mathscr{W}$,
$\mathrm{Psh}(\mathscr{W})$ models **DTT, with a universe** U^{HS}.

Let $W \in \mathrm{Obj}(\mathscr{W})$.
A pro W**-cell** of U^{HS} contains:

- a notion of dependent W-cells
- for all $V \in \mathrm{Obj}(\mathscr{W}/W)$, a notion of dependent V-cells

Looking at this differently

Define $\dot{\mathscr{W}} :\cong \mathscr{W}$.
If $W \in \mathrm{Obj}(\mathscr{W})$, then pro $W \in \mathrm{Obj}(\dot{\mathscr{W}})$.

Consider $\mathsf{U}^{\mathsf{HS}}_{\mathscr{W}} \in \mathrm{Psh}(\dot{\mathscr{W}})$.

pro W-cells have a different meaning:

- **Param’ty:** U of discrete types is not discrete.
  → Edges express het. equality; pro-edges express **relations**.
- **Directed:** U of Segal types is not Segal.
  → Arrows express morphisms; pro-arrows express **profunctors**.
  → Triangles express commutativity; pro-triangles are **boundary predicates**.

"Equipment pro-profunctors"!? Are you making this up?

✔ Only partially.

Hofmann-Streicher Universe [HS97]

For any category $\mathscr{W}$,
$\mathrm{Psh}(\mathscr{W})$ models **DTT, with a universe U^{HS}.**

Let $W \in \mathrm{Obj}(\mathscr{W})$.
A pro W**-cell** of U^{HS} contains:

- a notion of dependent W-cells
- for all $V \in \mathrm{Obj}(\mathscr{W}/W)$, a notion of dependent V-cells

Looking at this differently

Define $\dot{\mathscr{W}} :\cong \mathscr{W}$.
If $W \in \mathrm{Obj}(\mathscr{W})$, then pro $W \in \mathrm{Obj}(\dot{\mathscr{W}})$.

Consider $\mathrm{U}^{\mathrm{HS}}_{\mathscr{W}} \in \mathrm{Psh}(\dot{\mathscr{W}})$.

pro W-cells have a different meaning:

- **Param'ty:** U of discrete types is not discrete.
  ➔ Edges express het. equality;
  pro-edges express **relations**.
- **Directed:** U of Segal types is not Segal.
  ➔ Arrows express morphisms;
  pro-arrows express **profunctors**.
  ➔ Triangles express commutativity;
  pro-triangles are **boundary predicates**.

“Equipment pro-profunctors”!? Are you making this up?

✔ Only partially.

Hofmann-Streicher Universe [HS97]

For any category $\mathscr{W}$,
$\mathrm{Psh}(\mathscr{W})$ models **DTT, with a universe U^{HS}.**

Let $W \in \mathrm{Obj}(\mathscr{W})$.
A pro W**-cell** of U^{HS} contains:

- a notion of dependent W-cells
- for all $V \in \mathrm{Obj}(\mathscr{W}/W)$, a notion of dependent V-cells

Looking at this differently

Define $\dot{\mathscr{W}} :\cong \mathscr{W}$.
If $W \in \mathrm{Obj}(\mathscr{W})$, then pro $W \in \mathrm{Obj}(\dot{\mathscr{W}})$.

Consider $\mathsf{U}^{\mathsf{HS}}_{\mathscr{W}} \in \mathrm{Psh}(\dot{\mathscr{W}})$.

pro W-cells have a different meaning:

- **Param’ty:** U of discrete types is not discrete.
  ➔ Edges express het. equality; pro-edges express **relations**.
- **Directed:** U of Segal types is not Segal.
  ➔ Arrows express morphisms; pro-arrows express **profunctors**.
  ➔ Triangles express commutativity; pro-triangles are **boundary predicates**.

"Equipment pro-profunctors"!? Are you making this up?

✓ Only partially.

Hofmann-Streicher Universe [HS97]

For any category $\mathscr{W}$,
$\mathrm{Psh}(\mathscr{W})$ models **DTT, with a universe** U^{HS}.

Let $W \in \mathrm{Obj}(\mathscr{W})$.
A pro W**-cell** of U^{HS} contains:

- a notion of dependent W-cells
- for all $V \in \mathrm{Obj}(\mathscr{W}/W)$, a notion of dependent V-cells

Looking at this differently

Define $\dot{\mathscr{W}} :\cong \mathscr{W}$.
If $W \in \mathrm{Obj}(\mathscr{W})$, then pro $W \in \mathrm{Obj}(\dot{\mathscr{W}})$.

Consider $\mathsf{U}^{\mathsf{HS}}_{\mathscr{W}} \in \mathrm{Psh}(\dot{\mathscr{W}})$.

pro W-cells have a different meaning:

- **Param'ty:** U of discrete types is not discrete.
  ➔ Edges express het. equality;
  pro-edges express **relations**.
- **Directed:** U of Segal types is not Segal.
  ➔ Arrows express morphisms;
  pro-arrows express **profunctors**.
  ➔ Triangles express commutativity;
  pro-triangles are **boundary predicates**.

“Equipment pro-profunctors”!?
Are you making this up?

✔ Only partially.

Hofmann-Streicher Universe [HS97]

For any category $\mathscr{W}$,
$\mathrm{Psh}(\mathscr{W})$ models **DTT, with a universe** U^{HS}.

Let $W \in \mathrm{Obj}(\mathscr{W})$.
A pro W**-cell** of U^{HS} contains:

- a notion of dependent W-cells
- for all $V \in \mathrm{Obj}(\mathscr{W}/W)$,
  a notion of dependent V-cells

Looking at this differently

Define $\dot{\mathscr{W}} :\cong \mathscr{W}$.
If $W \in \mathrm{Obj}(\mathscr{W})$, then pro $W \in \mathrm{Obj}(\dot{\mathscr{W}})$.

Consider $\mathsf{U}^{\mathsf{HS}}_{\mathscr{W}} \in \mathrm{Psh}(\dot{\mathscr{W}})$.

pro W-cells have a different meaning:

- **Param'ty:** U of discrete types is not discrete.
  ➔ Edges express het. equality;
  pro-edges express **relations**.
- **Directed:** U of Segal types is not Segal.
  ➔ Arrows express morphisms;
  pro-arrows express **profunctors**.
  ➔ Triangles express commutativity;
  pro-triangles are **boundary predicates**.

“Equipment pro-profunctors”!? Are you making this up?

✔ Only partially.

Hofmann-Streicher Universe [HS97]

For any category $\mathscr{W}$,
$\mathrm{Psh}(\mathscr{W})$ models **DTT, with a universe U^{HS}.**

Let $W \in \mathrm{Obj}(\mathscr{W})$.
A pro W**-cell** of U^{HS} contains:

- a notion of dependent W-cells
- for all $V \in \mathrm{Obj}(\mathscr{W}/W)$, a notion of dependent V-cells

Looking at this differently

Define $\dot{\mathscr{W}} :\cong \mathscr{W}$.
If $W \in \mathrm{Obj}(\mathscr{W})$, then pro $W \in \mathrm{Obj}(\dot{\mathscr{W}})$.

Consider $\mathrm{U}^{\mathrm{HS}}_{\mathscr{W}} \in \mathrm{Psh}(\dot{\mathscr{W}})$.

$\mathrm{U}^{\mathrm{HS}}_{\mathscr{W}}$ is a **category** internal to $\mathrm{Psh}(\dot{\mathscr{W}})$

➔ $\mathrm{U}^{\mathrm{HS}}_{\mathscr{W}}$ is a **simplicial set** internal to $\mathrm{Psh}(\dot{\mathscr{W}})$

➔ $\mathrm{U}^{\mathrm{dir}}_{\mathscr{W}} \in \mathrm{Psh}(\dot{\mathscr{W}} \times \Delta)$.

☺ **Directed layer** on top of your favorite TT!

In particular:

		$\mathrm{Psh}(\top)$	(sets)
$\mathrm{U}^{\mathrm{dir}}_{\top}$	$\in$	$\mathrm{Psh}(\Delta)$	(categories)
$\mathrm{U}^{\mathrm{dir}}_{\Delta}$	$\in$	$\mathrm{Psh}(\dot{\Delta} \times \Delta)$	(eqmnts)
$\mathrm{U}^{\mathrm{dir}}_{\dot{\Delta}\times\Delta}$	$\in$	$\mathrm{Psh}(\ddot{\Delta} \times \dot{\Delta} \times \Delta)$	(2-eqmnts)

“Equipment pro-profunctors”!?
Are you making this up?

✔ Only partially.

Hofmann-Streicher Universe [HS97]

For any category $\mathscr{W}$,
$\mathrm{Psh}(\mathscr{W})$ models **DTT, with a universe U^{HS}.**

Let $W \in \mathrm{Obj}(\mathscr{W})$.
A pro W**-cell** of U^{HS} contains:

- a notion of dependent W-cells
- for all $V \in \mathrm{Obj}(\mathscr{W}/W)$,
  a notion of dependent V-cells

Looking at this differently

Define $\dot{\mathscr{W}} :\cong \mathscr{W}$.
If $W \in \mathrm{Obj}(\mathscr{W})$, then pro $W \in \mathrm{Obj}(\dot{\mathscr{W}})$.

Consider $\mathrm{U}^{\mathrm{HS}}_{\mathscr{W}} \in \mathrm{Psh}(\dot{\mathscr{W}})$.

$\mathrm{U}^{\mathrm{HS}}_{\mathscr{W}}$ is a **category** internal to $\mathrm{Psh}(\dot{\mathscr{W}})$

➔ $\mathrm{U}^{\mathrm{HS}}_{\mathscr{W}}$ is a **simplicial set** internal to $\mathrm{Psh}(\dot{\mathscr{W}})$

➔ $\mathrm{U}^{\mathrm{dir}}_{\mathscr{W}} \in \mathrm{Psh}(\dot{\mathscr{W}} \times \Delta)$.

☺ **Directed layer** on top of your favorite TT!

In particular:

		$\mathrm{Psh}(\top)$	(sets)
$\mathrm{U}^{\mathrm{dir}}_{\top}$	$\in$	$\mathrm{Psh}(\Delta)$	(categories)
$\mathrm{U}^{\mathrm{dir}}_{\Delta}$	$\in$	$\mathrm{Psh}(\dot{\Delta} \times \Delta)$	(eqmnts)
$\mathrm{U}^{\mathrm{dir}}_{\dot{\Delta}\times\Delta}$	$\in$	$\mathrm{Psh}(\ddot{\Delta} \times \dot{\Delta} \times \Delta)$	(2-eqmnts)

"Equipment pro-profunctors"!?
Are you making this up?

Only partially.

Hofmann-Streicher Universe [HS97]

For any category $\mathscr{W}$,
$\mathrm{Psh}(\mathscr{W})$ models **DTT, with a universe U^{HS}.**

Let $W \in \mathrm{Obj}(\mathscr{W})$.
A **pro W-cell** of U^{HS} contains:

- a notion of dependent W-cells
- for all $V \in \mathrm{Obj}(\mathscr{W}/W)$,
  a notion of dependent V-cells

Looking at this differently

Define $\dot{\mathscr{W}} :\cong \mathscr{W}$.
If $W \in \mathrm{Obj}(\mathscr{W})$, then pro $W \in \mathrm{Obj}(\dot{\mathscr{W}})$.

Consider $\mathsf{U}^{\mathsf{HS}}_{\mathscr{W}} \in \mathrm{Psh}(\dot{\mathscr{W}})$.

$\mathsf{U}^{\mathsf{HS}}_{\mathscr{W}}$ is a **category** internal to $\mathrm{Psh}(\mathscr{W})$
➔ $\mathsf{U}^{\mathsf{HS}}_{\mathscr{W}}$ is a **simplicial set** internal to $\mathrm{Psh}(\mathscr{W})$
➔ $\mathsf{U}^{\mathsf{dir}}_{\mathscr{W}} \in \mathrm{Psh}(\dot{\mathscr{W}} \times \Delta)$.
☺ **Directed layer** on top of your favorite TT!

In particular:

$$\begin{array}{lcll} & & \mathrm{Psh}(\top) & \text{(sets)} \\ \mathsf{U}^{\mathsf{dir}}_{\top} & \in & \mathrm{Psh}(\Delta) & \text{(categories)} \\ \mathsf{U}^{\mathsf{dir}}_{\Delta} & \in & \mathrm{Psh}(\dot{\Delta} \times \Delta) & \text{(eqmnts)} \\ \mathsf{U}^{\mathsf{dir}}_{\dot{\Delta}\times\Delta} & \in & \mathrm{Psh}(\ddot{\Delta} \times \dot{\Delta} \times \Delta) & \text{(2-eqmnts)} \end{array}$$

Against self-classification

By nature, classifiers (typically) do **NOT** contain themselves:

- **All of mankind** is **not** an example of a **human**.
- **The world's literature** is **not** an example of a **book**.

Forcing things to be otherwise is (a priori) **unreasonable**.

Classifiers of **collection-like** objects:

- **Set** is **more than** a (large) set.
- **Cat** is **more than** a (large) category.

It's not because you **can truncate** to achieve **self-classification**, that you **should!**

➜ Provide the user with the **unscathed classifier** *and* the **truncation modality**.
➜ Use **multimode** type theory.

☺ **Fixpoints:** ∞**Grpd** is a (large) ∞-groupoid.

Against self-classification

By nature, classifiers (typically) do **NOT** contain themselves:

- **All of mankind** is **not** an example of a **human**.
- **The world's literature** is **not** an example of a **book**.

Forcing things to be otherwise is (a priori) **unreasonable**.

Classifiers of **collection-like** objects:

- **Set** is **more than** a (large) set.
- **Cat** is **more than** a (large) category.

It's not because you **can truncate** to achieve **self-classification**, that you **should!**

➔ Provide the user with the **unscathed classifier** *and* the **truncation modality**.
➔ Use **multimode** type theory.

☺ **Fixpoints:** ∞**Grpd** is a (large) ∞-groupoid.

Against self-classification

By nature, classifiers (typically) do **NOT** contain themselves:

- **All of mankind** is **not** an example of a **human**.
- **The world's literature** is **not** an example of a **book**.

Forcing things to be otherwise is (a priori) **unreasonable**.

Classifiers of **collection-like** objects:

- **Set** is **more than** a (large) set.
- **Cat** is **more than** a (large) category.

It's not because you **can truncate** to achieve **self-classification**, that you **should!**

➔ Provide the user with the **unscathed classifier** *and* the **truncation modality**.
➔ Use **multimode** type theory.

☺ **Fixpoints:** ∞**Grpd** is a (large) ∞-groupoid.

Against self-classification

By nature, classifiers (typically) do **NOT** contain themselves:

- **All of mankind** is **not** an example of a **human**.
- **The world's literature** is **not** an example of a **book**.

Forcing things to be otherwise is (a priori) **unreasonable**.

Classifiers of **collection-like** objects:

- **Set** is **more than** a (large) set.
- **Cat** is **more than** a (large) category.

It's not because you **can truncate** to achieve **self-classification**, that you **should!**

➔ Provide the user with the **unscathed classifier** *and* the **truncation modality**.
➔ Use **multimode** type theory.

☺ **Fixpoints:** ∞**Grpd** is a (large) ∞-groupoid.

. . . and while I am ranting . . .

Against the Grothendieck Construction as "the" Σ-type for categories

Grothendieck Construction

Given a **category** $\mathscr{C}$ and
a **functor** $\mathscr{D} : \mathscr{C} \to \mathsf{Arws}(\mathsf{Cat})$,
i.e. **eqmnt functor** $\mathscr{D}' : \mathsf{FPro}(\mathscr{C}) \to \mathsf{Cat}$,
the category $\int_{\mathscr{C}} \mathscr{D}$ has:

- objects $(c, d \in \mathscr{D}(c))$
- morphisms

$$\left(c_1 \xrightarrow{\gamma} c_2, \mathscr{D}(\gamma)(d_1) \xrightarrow{\delta} d_2\right)$$

$\mathsf{Arws}(\mathsf{Cat}) \in \mathsf{Cat}$ is **truncated**.

$$\mathsf{FPro} \dashv \mathsf{Arws} : \mathsf{Eqmnt} \to \mathsf{Cat}$$

Arws	Discards pro-arrows
FPro	Freely adds "graph" pro-arrows
Pros	Discards arrows

Let's generalize from $\mathsf{FPro}(\mathscr{C})$ to $\mathscr{C}' \in \mathsf{Eqmnt}$.

$$\begin{tikzcd} \int_{\mathscr{C}} \mathscr{D} & \mathsf{Pros}\left(\oint_{\mathsf{FPro}(\mathscr{C})} \mathscr{D}'\right) \arrow[d, "\mathsf{Pros}(\mathsf{Fst})"'] \\ \mathscr{C} & \mathsf{Pros}(\mathsf{FPro}(\mathscr{C})) \end{tikzcd}$$

Against the Grothendieck Construction as “the” Σ-type for categories

Grothendieck Construction

Given a **category** $\mathscr{C}$ and
a **functor** $\mathscr{D} : \mathscr{C} \to \mathsf{Arws}(\mathsf{Cat})$,
i.e. **eqmnt functor** $\mathscr{D}' : \mathsf{FPro}(\mathscr{C}) \to \mathsf{Cat}$,
the category $\int_{\mathscr{C}} \mathscr{D}$ has:

- objects $(c, d \in \mathscr{D}(c))$
- morphisms

$$\left(c_1 \xrightarrow{\gamma} c_2, \mathscr{D}(\gamma)(d_1) \xrightarrow{\delta} d_2\right)$$

$\mathsf{Arws}(\mathsf{Cat}) \in \mathsf{Cat}$ is **truncated**.

$$\mathsf{FPro} \dashv \mathsf{Arws} : \mathsf{Eqmnt} \to \mathsf{Cat}$$

Arws	Discards pro-arrows
FPro	Freely adds “graph” pro-arrows
Pros	Discards arrows

Let’s generalize from $\mathsf{FPro}(\mathscr{C})$ to $\mathscr{C}' \in \mathsf{Eqmnt}$.

$$\begin{tikzcd}
\int_{\mathscr{C}} \mathscr{D} & \mathsf{Pros}\left(\oint_{\mathsf{FPro}(\mathscr{C})} \mathscr{D}'\right) \arrow[d, "\mathsf{Pros}(\mathsf{Fst})"'] \\
\mathscr{C} & \mathsf{Pros}(\mathsf{FPro}(\mathscr{C}))
\end{tikzcd}$$

Against the Grothendieck Construction as “**the**” Σ-type for categories

Grothendieck Construction

Given a **category** $\mathscr{C}$ and
a **functor** $\mathscr{D} : \mathscr{C} \to \mathrm{Arws}(\mathrm{Cat})$,
i.e. **eqmnt functor** $\mathscr{D}' : \mathrm{FPro}(\mathscr{C}) \to \mathrm{Cat}$,
the category $\int_{\mathscr{C}} \mathscr{D}$ has:

- objects $(c, d \in \mathscr{D}(c))$
- morphisms

$$\left(c_1 \xrightarrow{\gamma} c_2, \mathscr{D}(\gamma)(d_1) \xrightarrow{\delta} d_2\right)$$

$\mathrm{Arws}(\mathrm{Cat}) \in \mathrm{Cat}$ is **truncated**.

$$\mathrm{FPro} \dashv \mathrm{Arws} : \mathrm{Eqmnt} \to \mathrm{Cat}$$

Arws	Discards pro-arrows
FPro	Freely adds “graph” pro-arrows
Pros	Discards arrows

Let’s generalize from $\mathrm{FPro}(\mathscr{C})$ to $\mathscr{C}' \in \mathrm{Eqmnt}$.

$$\begin{tikzcd}
\int_{\mathscr{C}} \mathscr{D} & \mathrm{Pros}\left(\oint_{\mathrm{FPro}(\mathscr{C})} \mathscr{D}'\right) \arrow[d, "\mathrm{Pros}(\mathrm{Fst})"'] \\
\mathscr{C} & \mathrm{Pros}(\mathrm{FPro}(\mathscr{C}))
\end{tikzcd}$$

Against the Grothendieck Construction as "the" Σ-type for categories

Grothendieck Construction

Given a **category** $\mathscr{C}$ and
a **functor** $\mathscr{D} : \mathscr{C} \to \mathrm{Arws}(\mathrm{Cat})$,
i.e. **eqmnt functor** $\mathscr{D}' : \mathrm{FPro}(\mathscr{C}) \to \mathrm{Cat}$,
the category $\int_{\mathscr{C}} \mathscr{D}$ has:

- objects $(c, d \in \mathscr{D}(c))$
- morphisms

$$\left(c_1 \xrightarrow{\gamma} c_2, \mathscr{D}(\gamma)(d_1) \xrightarrow{\delta} d_2\right)$$

$\mathrm{Arws}(\mathrm{Cat}) \in \mathrm{Cat}$ is **truncated**.

$$\mathrm{FPro} \dashv \mathrm{Arws} : \mathrm{Eqmnt} \to \mathrm{Cat}$$

Arws	Discards pro-arrows
FPro	Freely adds "graph" pro-arrows
Pros	Discards arrows

Let's generalize from $\mathrm{FPro}(\mathscr{C})$ to $\mathscr{C}' \in \mathrm{Eqmnt}$.

$$\begin{tikzcd}
\int_{\mathscr{C}} \mathscr{D} & \mathrm{Pros}\left(\oint_{\mathrm{FPro}(\mathscr{C})} \mathscr{D}'\right) \arrow[d, "\mathrm{Pros}(\mathrm{Fst})"'] \\
\mathscr{C} & \mathrm{Pros}(\mathrm{FPro}(\mathscr{C}))
\end{tikzcd}$$

Against the Grothendieck Construction as "**the**" Σ-type for categories

Grothendieck Construction

Given a **category** $\mathscr{C}$ and
a **functor** $\mathscr{D} : \mathscr{C} \to \mathrm{Arws}(\mathrm{Cat})$,
i.e. **eqmnt functor** $\mathscr{D}' : \mathrm{FPro}(\mathscr{C}) \to \mathrm{Cat}$,
the category $\int_{\mathscr{C}} \mathscr{D}$ has:

- objects $(c, d \in \mathscr{D}(c))$
- morphisms

$$\left(c_1 \xrightarrow{\gamma} c_2, \mathscr{D}(\gamma)(d_1) \xrightarrow{\delta} d_2\right)$$

$\mathrm{Arws}(\mathrm{Cat}) \in \mathrm{Cat}$ is **truncated**.

$$\mathrm{FPro} \dashv \mathrm{Arws} : \mathrm{Eqmnt} \to \mathrm{Cat}$$

Arws	Discards pro-arrows
FPro	Freely adds "graph" pro-arrows
Pros	Discards arrows

Let's generalize from $\mathrm{FPro}(\mathscr{C})$ to $\mathscr{C}' \in \mathrm{Eqmnt}$.

$$\begin{tikzcd}
\int_{\mathscr{C}} \mathscr{D} & \mathrm{Pros}\left(\oint_{\mathrm{FPro}(\mathscr{C})} \mathscr{D}'\right) \arrow[d, "\mathrm{Pros}(\mathrm{Fst})"'] \\
\mathscr{C} & \mathrm{Pros}(\mathrm{FPro}(\mathscr{C}))
\end{tikzcd}$$

Against the Grothendieck Construction as “the” Σ-type for categories

Grothendieck Construction

Given a **category** $\mathscr{C}$ and
a **functor** $\mathscr{D} : \mathscr{C} \to \mathrm{Arws}(\mathrm{Cat})$,
i.e. **eqmnt functor** $\mathscr{D}' : \mathrm{FPro}(\mathscr{C}) \to \mathrm{Cat}$,
the category $\int_{\mathscr{C}} \mathscr{D}$ has:

- objects $(c, d \in \mathscr{D}(c))$
- morphisms

$$\left(c_1 \xrightarrow{\gamma} c_2, \mathscr{D}(\gamma)(d_1) \xrightarrow{\delta} d_2\right)$$

$\mathrm{Arws}(\mathrm{Cat}) \in \mathrm{Cat}$ is **truncated**.

$$\mathrm{FPro} \dashv \mathrm{Arws} : \mathrm{Eqmnt} \to \mathrm{Cat}$$

Arws	Discards pro-arrows
FPro	Freely adds “graph” pro-arrows
Pros	Discards arrows

Let’s generalize from $\mathrm{FPro}(\mathscr{C})$ to $\mathscr{C}' \in \mathrm{Eqmnt}$.

$$\begin{tikzcd}
\int_{\mathscr{C}} \mathscr{D} & \mathrm{Pros}\left(\oint_{\mathrm{FPro}(\mathscr{C})} \mathscr{D}'\right) \arrow[d, "\mathrm{Pros}(\mathrm{Fst})"'] \\
\mathscr{C} & \mathrm{Pros}(\mathrm{FPro}(\mathscr{C}))
\end{tikzcd}$$

Grothendieck Construction

Given a **category** $\mathscr{C}$ and
a **functor** $\mathscr{D} : \mathscr{C} \to \mathrm{Arws}(\mathrm{Cat})$,
i.e. **eqmnt functor** $\mathscr{D}' : \mathrm{FPro}(\mathscr{C}) \to \mathrm{Cat}$,
the category $\int_{\mathscr{C}} \mathscr{D}$ has:

- objects $(c, d \in \mathscr{D}(c))$
- morphisms

$$\left(c_1 \xrightarrow{\gamma} c_2, \mathscr{D}(\gamma)(d_1) \xrightarrow{\delta} d_2\right)$$

$\mathrm{Arws}(\mathrm{Cat}) \in \mathrm{Cat}$ is **truncated**.

$$\mathrm{FPro} \dashv \mathrm{Arws} : \mathrm{Eqmnt} \to \mathrm{Cat}$$

Arws	Discards pro-arrows
FPro	Freely adds “graph” pro-arrows
Pros	Discards arrows

Let’s generalize from $\mathrm{FPro}(\mathscr{C})$ to $\mathscr{C}' \in \mathrm{Eqmnt}$.

$$\begin{tikzcd}
\int_{\mathscr{C}} \mathscr{D} & \mathrm{Pros}\left(\oint_{\mathrm{FPro}(\mathscr{C})} \mathscr{D}'\right) \arrow[d, "\mathrm{Pros}(\mathrm{Fst})"'] \\
\mathscr{C} & \mathrm{Pros}(\mathrm{FPro}(\mathscr{C}))
\end{tikzcd}$$

Against the Grothendieck Construction as “the” Σ-type for categories

Grothendieck Construction

Given a **category** $\mathscr{C}$ and
a **functor** $\mathscr{D} : \mathscr{C} \to \mathrm{Arws}(\mathrm{Cat})$,
i.e. **eqmnt functor** $\mathscr{D}' : \mathrm{FPro}(\mathscr{C}) \to \mathrm{Cat}$,
the category $\int_{\mathscr{C}} \mathscr{D}$ has:

- objects $(c, d \in \mathscr{D}(c))$
- morphisms

$$\left(c_1 \xrightarrow{\gamma} c_2, \mathscr{D}(\gamma)(d_1) \xrightarrow{\delta} d_2\right)$$

$\mathrm{Arws}(\mathrm{Cat}) \in \mathrm{Cat}$ is **truncated**.

$$\mathrm{FPro} \dashv \mathrm{Arws} : \mathrm{Eqmnt} \to \mathrm{Cat}$$

Arws	Discards pro-arrows
FPro	Freely adds “graph” pro-arrows
Pros	Discards arrows

Let’s generalize from $\mathrm{FPro}(\mathscr{C})$ to $\mathscr{C}' \in \mathrm{Eqmnt}$.

$$\begin{tikzcd}
\int_{\mathscr{C}} \mathscr{D} & \mathrm{Pros}\left(\oint_{\mathrm{FPro}(\mathscr{C})} \mathscr{D}'\right) \arrow[d, "\mathrm{Pros}(\mathrm{Fst})"'] \\
\mathscr{C} & \mathrm{Pros}(\mathrm{FPro}(\mathscr{C}))
\end{tikzcd}$$

Against the Grothendieck Construction as "the" Σ-type for categories

"Equipment of elements"

Given an **eqmnt** $\mathscr{C}'$ and
an **eqmnt functor** $\mathscr{D}' : \mathscr{C}' \to \mathrm{Cat}$,
the category $\oint_{\mathscr{C}'} \mathscr{D}'$ has:

- objects $(c, d \in \mathscr{D}'(c))$
- morphisms

$$\left(c_1 \xrightarrow{\gamma} c_2, d_1 \mapsto_{\mathscr{D}'(\gamma)} d_2\right)$$

- pro-arrows

$$\left(c_1 \overset{p}{\nrightarrow} c_2, d_1 \xrightarrow{\delta}_{\mathscr{D}'(p)} d_2\right)$$

$\approx$ cat. of elements internal to $\mathrm{Psh}(\Delta)$

Arws(Cat) $\in$ Cat is **truncated**.

$$\mathrm{FPro} \dashv \mathrm{Arws} : \mathrm{Eqmnt} \to \mathrm{Cat}$$

Arws	Discards pro-arrows
FPro	Freely adds "graph" pro-arrows
Pros	Discards arrows

Let's generalize from $\mathrm{FPro}(\mathscr{C})$ to $\mathscr{C}' \in \mathrm{Eqmnt}$.

$$\begin{tikzcd}
\int_{\mathscr{C}} \mathscr{D} & \mathrm{Pros}\left(\oint_{\mathrm{FPro}(\mathscr{C})} \mathscr{D}'\right) \arrow[d, "\mathrm{Pros}(\mathrm{Fst})"'] \\
\mathscr{C} & \mathrm{Pros}(\mathrm{FPro}(\mathscr{C}))
\end{tikzcd}$$

“Equipment of elements”

Given an **eqmnt** $\mathscr{C}'$ and
an **eqmnt functor** $\mathscr{D}' : \mathscr{C}' \to \mathsf{Cat}$,
the category $\oint_{\mathscr{C}'} \mathscr{D}'$ has:

- objects $(c, d \in \mathscr{D}'(c))$
- morphisms

$$\left(c_1 \xrightarrow{\gamma} c_2, d_1 \mapsto_{\mathscr{D}'(\gamma)} d_2\right)$$

- pro-arrows

$$\left(c_1 \overset{p}{\nrightarrow} c_2, d_1 \xrightarrow{\delta}_{\mathscr{D}'(p)} d_2\right)$$

$\approx$ cat. of elements internal to $\mathrm{Psh}(\Delta)$

$\mathsf{Arws}(\mathsf{Cat}) \in \mathsf{Cat}$ is **truncated**.

$$\mathsf{FPro} \dashv \mathsf{Arws} : \mathsf{Eqmnt} \to \mathsf{Cat}$$

Arws	Discards pro-arrows
FPro	Freely adds “graph” pro-arrows
Pros	Discards arrows

Let’s generalize from $\mathsf{FPro}(\mathscr{C})$ to $\mathscr{C}' \in \mathsf{Eqmnt}$.

$$\begin{tikzcd} \int_{\mathscr{C}} \mathscr{D} & \mathsf{Pros}\left(\oint_{\mathsf{FPro}(\mathscr{C})} \mathscr{D}'\right) \arrow[d, "\mathsf{Pros}(\mathsf{Fst})"'] \\ \mathscr{C} & \mathsf{Pros}(\mathsf{FPro}(\mathscr{C})) \end{tikzcd}$$

Against the Grothendieck Construction as "the" Σ-type for categories

"Equipment of elements"

Given an **eqmnt** $\mathscr{C}'$ and
an **eqmnt functor** $\mathscr{D}' : \mathscr{C}' \to \mathrm{Cat}$,
the category $\oint_{\mathscr{C}'} \mathscr{D}'$ has:

- objects $(c, d \in \mathscr{D}'(c))$
- morphisms

$$\left(c_1 \xrightarrow{\gamma} c_2, d_1 \mapsto_{\mathscr{D}'(\gamma)} d_2\right)$$

- pro-arrows

$$\left(c_1 \overset{p}{\nrightarrow} c_2, d_1 \xrightarrow{\delta}_{\mathscr{D}'(p)} d_2\right)$$

$\approx$ cat. of elements internal to $\mathrm{Psh}(\Delta)$

Arws(Cat) $\in$ Cat is **truncated**.

$$\mathrm{FPro} \dashv \mathrm{Arws} : \mathrm{Eqmnt} \to \mathrm{Cat}$$

Arws	Discards pro-arrows
FPro	Freely adds "graph" pro-arrows
Pros	Discards arrows

Let's generalize from FPro($\mathscr{C}$) to $\mathscr{C}' \in$ Eqmnt.

$$\begin{tikzcd}
\int_{\mathscr{C}} \mathscr{D} & \mathrm{Pros}\left(\oint_{\mathrm{FPro}(\mathscr{C})} \mathscr{D}'\right) \arrow[d, "\mathrm{Pros}(\mathrm{Fst})"'] \\
\mathscr{C} & \mathrm{Pros}(\mathrm{FPro}(\mathscr{C}))
\end{tikzcd}$$

Against the Grothendieck Construction as “the” Σ-type for categories

“Equipment of elements”

Given an **eqmnt** $\mathscr{C}'$ and
an **eqmnt functor** $\mathscr{D}' : \mathscr{C}' \to \mathrm{Cat}$,
the category $\oint_{\mathscr{C}'} \mathscr{D}'$ has:

- objects $(c, d \in \mathscr{D}'(c))$
- morphisms

$$\left(c_1 \xrightarrow{\gamma} c_2, d_1 \mapsto_{\mathscr{D}'(\gamma)} d_2\right)$$

- pro-arrows

$$\left(c_1 \overset{p}{\nrightarrow} c_2, d_1 \xrightarrow{\delta}_{\mathscr{D}'(p)} d_2\right)$$

$\approx$ cat. of elements internal to $\mathrm{Psh}(\Delta)$

$\mathrm{Arws}(\mathrm{Cat}) \in \mathrm{Cat}$ is **truncated**.

$$\mathrm{FPro} \dashv \mathrm{Arws} : \mathrm{Eqmnt} \to \mathrm{Cat}$$

Arws	Discards pro-arrows
FPro	Freely adds “graph” pro-arrows
Pros	Discards arrows

Let’s generalize from $\mathrm{FPro}(\mathscr{C})$ to $\mathscr{C}' \in \mathrm{Eqmnt}$.

$$\int_{\mathscr{C}} \mathscr{D}$$

$$\begin{tikzcd}
 & \mathrm{Pros}\left(\oint_{\mathrm{FPro}(\mathscr{C})} \mathscr{D}'\right) \arrow[d, "\mathrm{Pros}(\mathrm{Fst})"] \\
\mathscr{C} \arrow[r, "\varphi \mapsto \varphi^{\ddagger}"] & \mathrm{Pros}(\mathrm{FPro}(\mathscr{C}))
\end{tikzcd}$$

Against the Grothendieck Construction as "the" Σ-type for categories

"Equipment of elements"

Given an **eqmnt** $\mathscr{C}'$ and
an **eqmnt functor** $\mathscr{D}' : \mathscr{C}' \to \mathsf{Cat}$,
the category $\oint_{\mathscr{C}'} \mathscr{D}'$ has:

- objects $(c, d \in \mathscr{D}'(c))$
- morphisms

$$\left(c_1 \xrightarrow{\gamma} c_2, d_1 \mapsto_{\mathscr{D}'(\gamma)} d_2\right)$$

- pro-arrows

$$\left(c_1 \overset{p}{\nrightarrow} c_2, d_1 \xrightarrow{\delta}_{\mathscr{D}'(p)} d_2\right)$$

$\approx$ cat. of elements internal to $\mathrm{Psh}(\Delta)$

$\mathsf{Arws}(\mathsf{Cat}) \in \mathsf{Cat}$ is **truncated**.

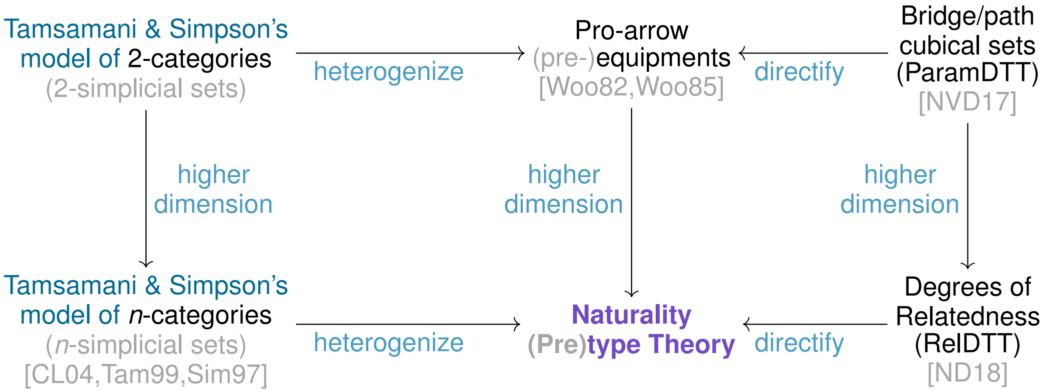
$$\mathsf{FPro} \dashv \mathsf{Arws} : \mathsf{Eqmnt} \to \mathsf{Cat}$$

Arws	Discards pro-arrows
FPro	Freely adds "graph" pro-arrows
Pros	Discards arrows

Let's generalize from $\mathsf{FPro}(\mathscr{C})$ to $\mathscr{C}' \in \mathsf{Eqmnt}$.

$$\begin{tikzcd}
\int_{\mathscr{C}} \mathscr{D} \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathsf{Pros}(\oint_{\mathsf{FPro}(\mathscr{C})} \mathscr{D}') \arrow[d, "\mathsf{Pros}(\mathsf{Fst})"'] \\
\mathscr{C} \arrow[r, "\varphi \mapsto \varphi^{\ddagger}"] & \mathsf{Pros}(\mathsf{FPro}(\mathscr{C}))
\end{tikzcd}$$

Three Approaches to the Model



Degrees of Relatedness
(RelDTT)

Nuyts and Devriese (2018) @ LICS

- **Relational version** of what NatTT intends to be
- Perhaps **alienating**:
  - Goes beyond Reynolds' parametricity
  - Much less than higher category theory
- Explains several **known relational modalities**
- Has the **virtue of existence**

Degrees of Relatedness (RelDTT)

Nuyts and Devriese (2018) @ LICS

- **Relational version** of what NatTT intends to be
- Perhaps **alienating**:
  - Goes beyond Reynolds' parametricity
  - Much less than higher category theory
- Explains several **known relational modalities**
- Has the **virtue of existence**

Degrees of Relatedness: Overview

- Parametricity is about **relations**,
- Equip types with **multiple**, **proof-relevant relations** $s \frown_i t$ indexed by **degree** i:
  - Just **one** for **small types** (Bool, $\mathbb{N} \to \mathbb{N}$, ...),
  - **More** for **larger** types ($\mathsf{U}_0 \to \mathsf{U}_0$, Grp, ...).
  - Proofs called ***i*-edges**.
- Describe **function behaviour** by saying how functions **influence degree** of relatedness,
- This explains
  - **parametricity**
  - **ad hoc polymorphism**
  - **. irrelevance**
  - **.. shape-irrelevance**
  - aspects of **algebra, unions, intersections, Prop, ...**

Degrees of Relatedness: Overview

- Parametricity is about **relations**,
- Equip types with multiple, **proof-relevant relations** $s \frown_i t$ indexed by **degree** i:
  - Just **one** for **small types** (Bool, $\mathbb{N} \to \mathbb{N}, \ldots$),
  - **More** for **larger** types ($U_0 \to U_0$, Grp, $\ldots$).
  - Proofs called i-**edges**.
- Describe **function behaviour** by saying how functions **influence degree** of relatedness,
- This explains
  - **parametricity**
  - **ad hoc polymorphism**
  - **. irrelevance**
  - **.. shape-irrelevance**
  - aspects of **algebra, unions, intersections, Prop, . . .**

Degrees of Relatedness: Overview

- Parametricity is about **relations**,
- Equip types with **multiple, proof-relevant relations** $s \frown_i t$ indexed by **degree** i:
  - Just **one** for **small types** (Bool, $\mathbb{N} \to \mathbb{N}$, ...),
  - **More** for **larger** types ($\mathsf{U}_0 \to \mathsf{U}_0$, Grp, ...).
  - Proofs called i**-edges**.
- Describe **function behaviour** by saying how functions **influence degree** of relatedness,
- This explains
  - **parametricity**
  - **ad hoc polymorphism**
  - **. irrelevance**
  - **.. shape-irrelevance**
  - aspects of **algebra, unions, intersections, Prop, ...**

Degrees of Relatedness: Overview

- Parametricity is about **relations**,
- Equip types with **multiple, proof-relevant relations** $s \frown_i t$ indexed by **degree** i:
  - Just **one** for **small types** (Bool, $\mathbb{N} \to \mathbb{N}$, ...),
  - **More** for **larger** types ($\mathsf{U}_0 \to \mathsf{U}_0$, Grp, ...).
  - Proofs called ***i*-edges**.
- Describe **function behaviour** by saying how functions **influence degree** of relatedness,
- This explains
  - **parametricity**
  - **ad hoc polymorphism**
  - **. irrelevance**
  - **.. shape-irrelevance**
  - aspects of **algebra, unions, intersections, Prop, ...**

Degrees of Relatedness: Overview

- Parametricity is about **relations**,
- Equip types with **multiple, proof-relevant relations** $s \frown_i t$ indexed by **degree** i:
  - Just **one** for **small types** (Bool, $\mathbb{N} \to \mathbb{N}$, ...),
  - **More** for **larger** types ($U_0 \to U_0$, Grp, ...).
  - Proofs called i-**edges**.
- Describe **function behaviour** by saying how functions **influence degree** of relatedness,
- This explains
  - **parametricity**
  - **ad hoc polymorphism**
  - **. irrelevance**
  - **.. shape-irrelevance**
  - aspects of **algebra, unions, intersections, Prop, ...**

Degrees of Relatedness: Overview

- Parametricity is about **relations**,
- Equip types with **multiple, proof-relevant relations** $s \frown_i t$ indexed by **degree** i:
  - Just **one** for **small types** (Bool, $\mathbb{N} \to \mathbb{N}, \ldots$),
  - **More** for **larger** types ($\mathsf{U}_0 \to \mathsf{U}_0$, Grp, $\ldots$).
  - Proofs called ***i*-edges**.
- Describe **function behaviour** by saying how functions **influence degree** of relatedness,
- This explains
  - **parametricity**
  - **ad hoc polymorphism**
  - **. irrelevance**
  - **.. shape-irrelevance**
  - aspects of **algebra, unions, intersections, Prop, …**

Degrees of Relatedness: Overview

- Parametricity is about **relations**,
- Equip types with **multiple, proof-relevant relations** $s \frown_i t$ indexed by **degree** i:
  - Just **one** for **small types** (Bool, $\mathbb{N} \to \mathbb{N}, \dots$),
  - **More** for **larger** types ($\mathsf{U}_0 \to \mathsf{U}_0$, Grp, $\dots$).
  - Proofs called ***i*-edges**.
- Describe **function behaviour** by saying how functions **influence degree** of relatedness,
- This explains
  - **parametricity**
  - **ad hoc polymorphism**
  - **. irrelevance**
  - **.. shape-irrelevance**
  - aspects of **algebra, unions, intersections, Prop, . . .**

Degrees of Relatedness: Overview

- Parametricity is about **relations**,
- Equip types with **multiple, proof-relevant relations** $s \frown_i t$ indexed by **degree** i:
  - Just **one** for **small types** (Bool, $\mathbb{N} \to \mathbb{N}, \ldots$),
  - **More** for **larger** types ($\mathsf{U}_0 \to \mathsf{U}_0$, Grp, $\ldots$).
  - Proofs called ***i*-edges**.
- Describe **function behaviour** by saying how functions **influence degree** of relatedness,
- This explains
  - **parametricity**
  - **ad hoc polymorphism**
  - **. irrelevance**
  - **.. shape-irrelevance**
  - aspects of **algebra, unions, intersections, Prop, . . .**

Degrees of Relatedness: Four Laws

- **Reflexivity:** $(a : A) \frown_i^A (a : A)$
  (Semantically, prop. eq. = def. eq.)
- **Degradation:** $((a : A) \frown_i^R (b : B)) \to ((a : A) \frown_{i+1}^R (b : B))$
- **Dependency:** $(a : A) \frown_i^R (b : B)$ **presumes** $R : A \frown_{i+1}^{\mathsf{U}} B$
- **Identity extension:** $(a : A) \frown_0^A (b : A)$ means $a = b : A$.
  $\rightsquigarrow$ heterogeneous $\frown_0$ serves as heterogeneous equality.

Degrees of Relatedness: Four Laws

- **Reflexivity:** $(a = b : A) \to (a : A) \frown^A_i (b : A)$
  (Semantically, prop. eq. = def. eq.)
- **Degradation:** $((a : A) \frown^R_i (b : B)) \to ((a : A) \frown^R_{i+1} (b : B))$
- **Dependency:** $(a : A) \frown^R_i (b : B)$ **presumes** $R : A \frown^{\mathrm{U}}_{i+1} B$
- **Identity extension:** $(a : A) \frown^A_0 (b : A)$ means $a = b : A$.
  $\rightsquigarrow$ heterogeneous $\frown_0$ serves as heterogeneous equality.

Degrees of Relatedness: Four Laws

- **Reflexivity:** $(a = b : A) \to (a : A) \frown_i^A (b : A)$
  (Semantically, prop. eq. = def. eq.)
- **Degradation:** $((a : A) \frown_i^R (b : B)) \to ((a : A) \frown_{i+1}^R (b : B))$
- **Dependency:** $(a : A) \frown_i^R (b : B)$ **presumes** $R : A \frown_{i+1}^{\mathrm{U}} B$
- **Identity extension:** $(a : A) \frown_0^A (b : A)$ means $a = b : A$.
  $\rightsquigarrow$ heterogeneous $\frown_0$ serves as heterogeneous equality.

Degrees of Relatedness: Four Laws

- **Reflexivity:** $(a = b : A) \to (a : A) \frown_i^A (b : A)$
  (Semantically, prop. eq. = def. eq.)
- **Degradation:** $((a : A) \frown_i^R (b : B)) \to ((a : A) \frown_{i+1}^R (b : B))$
- **Dependency:** $(a : A) \frown_i^R (b : B)$ **presumes** $R : A \frown_{i+1}^{\mathsf{U}} B$
- **Identity extension:** $(a : A) \frown_0^A (b : A)$ means $a = b : A$.
  $\rightsquigarrow$ heterogeneous $\frown_0$ serves as heterogeneous equality.

Degrees of Relatedness: Four Laws

- **Reflexivity:** $(a = b : A) \to (a : A) \frown_i^A (b : A)$
  (Semantically, prop. eq. = def. eq.)
- **Degradation:** $((a : A) \frown_i^R (b : B)) \to ((a : A) \frown_{i+1}^R (b : B))$
- **Dependency:** $(a : A) \frown_i^R (b : B)$ **presumes** $R : A \frown_{i+1}^{\mathsf{U}} B$
- **Identity extension:** $(a : A) \frown_0^A (b : A)$ means $a = b : A$.
  $\rightsquigarrow$ heterogeneous $\frown_0$ serves as heterogeneous equality.

Understanding degrees

$(a : A) \frown_0^A (b : A)$	Equality.
$(a : A) \frown_0^R (b : B)$	Heterogeneous equality along ...
$R : (A : \mathrm{U}^0) \frown_1^{\mathrm{U}^0} (B : \mathrm{U}^0)$	Any relation R.
$P : (G : \mathrm{Grp}) \frown_1^{\mathrm{Grp}} (H : \mathrm{Grp})$	Any logical/algebraic relation P.
$Q : (G : \mathrm{Grp}) \frown_1^V (M : \mathrm{Monoid})$	Any logical/algebraic relation Q along ...
$V : (\mathrm{Grp} : \mathrm{U}^1) \frown_2^{\mathrm{U}^1} (\mathrm{Monoid} : \mathrm{U}^1)$	V could specify that Q must respect e **and** $*$ (but it could ask Q to be anything).

Understanding degrees

$(a : A) \frown_0^A (b : A)$	Equality.
$(a : A) \frown_0^R (b : B)$	Heterogeneous equality along ...
$R : (A : \mathrm{U}^0) \frown_1^{\mathrm{U}^0} (B : \mathrm{U}^0)$	Any relation R.
$P : (G : \mathrm{Grp}) \frown_1^{\mathrm{Grp}} (H : \mathrm{Grp})$	Any logical/algebraic relation P.
$Q : (G : \mathrm{Grp}) \frown_1^V (M : \mathrm{Monoid})$	Any logical/algebraic relation Q along ...
$V : (\mathrm{Grp} : \mathrm{U}^1) \frown_2^{\mathrm{U}^1} (\mathrm{Monoid} : \mathrm{U}^1)$	V could specify that Q must respect e **and** $*$ (but it could ask Q to be anything).

Understanding degrees

$(a : A) \frown_0^{A} (b : A)$	Equality.
$(a : A) \frown_0^{R} (b : B)$	Heterogeneous equality along ...
$R : (A : \mathrm{U}^0) \frown_1^{\mathrm{U}^0} (B : \mathrm{U}^0)$	Any relation R.
$P : (G : \mathrm{Grp}) \frown_1^{\mathrm{Grp}} (H : \mathrm{Grp})$	Any logical/algebraic relation P.
$Q : (G : \mathrm{Grp}) \frown_1^{V} (M : \mathrm{Monoid})$	Any logical/algebraic relation Q along ...
$V : (\mathrm{Grp} : \mathrm{U}^1) \frown_2^{\mathrm{U}^1} (\mathrm{Monoid} : \mathrm{U}^1)$	V could specify that Q must respect e **and** $*$ (but it could ask Q to be anything).

Understanding degrees

$(a : A) \frown_0^A (b : A)$	Equality.
$(a : A) \frown_0^R (b : B)$	Heterogeneous equality along ...
$R : (A : \mathrm{U}^0) \frown_1^{\mathrm{U}^0} (B : \mathrm{U}^0)$	Any relation R.
$P : (G : \mathrm{Grp}) \frown_1^{\mathrm{Grp}} (H : \mathrm{Grp})$	Any logical/algebraic relation P.
$Q : (G : \mathrm{Grp}) \frown_1^V (M : \mathrm{Monoid})$	Any logical/algebraic relation Q along ...
$V : (\mathrm{Grp} : \mathrm{U}^1) \frown_2^{\mathrm{U}^1} (\mathrm{Monoid} : \mathrm{U}^1)$	V could specify that Q must respect e **and** $*$ (but it could ask Q to be anything).

Understanding degrees

$(a : A) \frown_0^A (b : A)$	Equality.
$(a : A) \frown_0^R (b : B)$	Heterogeneous equality along ...
$R : (A : \mathrm{U}^0) \frown_1^{\mathrm{U}^0} (B : \mathrm{U}^0)$	Any relation R.
$P : (G : \mathrm{Grp}) \frown_1^{\mathrm{Grp}} (H : \mathrm{Grp})$	Any logical/algebraic relation P.
$Q : (G : \mathrm{Grp}) \frown_1^V (M : \mathrm{Monoid})$	Any logical/algebraic relation Q along ...
$V : (\mathrm{Grp} : \mathrm{U}^1) \frown_2^{\mathrm{U}^1} (\mathrm{Monoid} : \mathrm{U}^1)$	V could specify that Q must respect e **and** $*$ (but it could ask Q to be anything).

Understanding degrees

$(a : A) \frown_0^A (b : A)$	Equality.
$(a : A) \frown_0^R (b : B)$	Heterogeneous equality along ...
$R : (A : \mathrm{U}^0) \frown_1^{\mathrm{U}^0} (B : \mathrm{U}^0)$	Any relation R.
$P : (G : \mathrm{Grp}) \frown_1^{\mathrm{Grp}} (H : \mathrm{Grp})$	Any logical/algebraic relation P.
$Q : (G : \mathrm{Grp}) \frown_1^V (M : \mathrm{Monoid})$	Any logical/algebraic relation Q along ...
$V : (\mathrm{Grp} : \mathrm{U}^1) \frown_2^{\mathrm{U}^1} (\mathrm{Monoid} : \mathrm{U}^1)$	V could specify that Q must respect e **and** $*$ (but it could ask Q to be anything).

Understanding degrees

$(a : A) \frown_0^A (b : A)$	Equality.
$(a : A) \frown_0^R (b : B)$	Heterogeneous equality along ...
$R : (A : \mathrm{U}^0) \frown_1^{\mathrm{U}^0} (B : \mathrm{U}^0)$	Any relation R.
$P : (G : \mathrm{Grp}) \frown_1^{\mathrm{Grp}} (H : \mathrm{Grp})$	Any logical/algebraic relation P.
$Q : (G : \mathrm{Grp}) \frown_1^V (M : \mathrm{Monoid})$	Any logical/algebraic relation Q along ...
$V : (\mathrm{Grp} : \mathrm{U}^1) \frown_2^{\mathrm{U}^1} (\mathrm{Monoid} : \mathrm{U}^1)$	V could specify that Q must respect e **and** $*$ (but it could ask Q to be anything).

Understanding modalities: Parametricity

par : types $\rightarrow$ values

$$if : (\textbf{par} \mid X : \mathrm{U}^0) \rightarrow B\,X$$

$$\begin{tikzcd}
X = Y \arrow[r] \arrow[d] & if_X = if_Y \arrow[dd] \\
R : X \frown^{\mathrm{U}^0}_{0} Y \arrow[d] & \\
R : X \frown^{\mathrm{U}^0}_{1} Y \arrow[r] \arrow[d] & if_X \frown^{B\,R}_{0} if_Y \arrow[d] \\
\top \arrow[r] & \top
\end{tikzcd}$$

con : types $\rightarrow$ types

$$B : \mathrm{U}^0 \rightarrow \mathrm{U}^0$$

$$B\,X = \mathrm{Bool} \rightarrow X \rightarrow X \rightarrow X$$

$$\begin{tikzcd}
X = Y \arrow[r] \arrow[d] & B\,X = B\,Y \arrow[d] \\
X \frown^{\mathrm{U}^0}_{0} Y \arrow[r, dotted, no head] \arrow[d] & B\,X \frown^{\mathrm{U}^0}_{0} B\,Y \arrow[d] \\
X \frown^{\mathrm{U}^0}_{1} Y \arrow[r, dotted, no head] \arrow[d] & B\,X \frown^{\mathrm{U}^0}_{1} B\,Y \arrow[d] \\
\top \arrow[r] & \top
\end{tikzcd}$$

par : types $\to$ values

$$\mathit{if} : (\mathbf{par} \mid X : \mathrm{U}^0) \to B\, X$$

$$\begin{tikzcd}
X = Y \arrow[rr] \arrow[d] & & \mathit{if}_X = \mathit{if}_Y \arrow[dd] \\
R : X \frown^{\mathrm{U}^0}_0 Y \arrow[d] & & \\
R : X \frown^{\mathrm{U}^0}_1 Y \arrow[r] \arrow[d] & \mathit{if}_X \frown^{B\,R}_0 \mathit{if}_Y \arrow[d, phantom] & \\
\top \arrow[rr] & & \top
\end{tikzcd}$$

$$\begin{tikzcd}
R : X \frown^{\mathrm{U}^0}_1 Y \arrow[r] & \mathit{if}_X \frown^{B\,R}_0 \mathit{if}_Y \arrow[d] \\
\top \arrow[r] & \top
\end{tikzcd}$$

con : types $\to$ types

$$B : \mathrm{U}^0 \to \mathrm{U}^0$$

$$B\, X = \mathrm{Bool} \to X \to X \to X$$

$$\begin{tikzcd}
X = Y \arrow[r] \arrow[d] & B\,X = B\,Y \arrow[d] \\
X \frown^{\mathrm{U}^0}_0 Y \arrow[r, dotted, no head] \arrow[d] & B\,X \frown^{\mathrm{U}^0}_0 B\,Y \arrow[d] \\
X \frown^{\mathrm{U}^0}_1 Y \arrow[r, dotted, no head] \arrow[d] & B\,X \frown^{\mathrm{U}^0}_1 B\,Y \arrow[d] \\
\top \arrow[r] & \top
\end{tikzcd}$$

Understanding modalities: Parametricity

par : types $\to$ values

$$\mathit{if} : (\textbf{par} \mid X : \mathrm{U}^0) \to B\,X$$

$$\begin{tikzcd}
X = Y \arrow[r] \arrow[d] & \mathit{if}_X = \mathit{if}_Y \arrow[dd] \\
R : X \frown^{\mathrm{U}^0}_0 Y \arrow[d] & \\
R : X \frown^{\mathrm{U}^0}_1 Y \arrow[r] \arrow[d] & \mathit{if}_X \frown^{B\,R}_0 \mathit{if}_Y \arrow[d] \\
\top \arrow[r] & \top
\end{tikzcd}$$

con : types $\to$ types

$$B : \mathrm{U}^0 \to \mathrm{U}^0$$

$$B\,X = \mathrm{Bool} \to X \to X \to X$$

$$\begin{tikzcd}
X = Y \arrow[r] \arrow[d] & B\,X = B\,Y \arrow[d] \\
X \frown^{\mathrm{U}^0}_0 Y \arrow[r, dotted, no head] \arrow[d] & B\,X \frown^{\mathrm{U}^0}_0 B\,Y \arrow[d] \\
X \frown^{\mathrm{U}^0}_1 Y \arrow[r, dotted, no head] \arrow[d] & B\,X \frown^{\mathrm{U}^0}_1 B\,Y \arrow[d] \\
\top \arrow[r] & \top
\end{tikzcd}$$

Understanding modalities: Parametricity

par : types $\to$ values

$$if : (\mathbf{par} \mid X : \mathrm{U}^0) \to B\,X$$

$$\begin{tikzcd}
X = Y \arrow[r] \arrow[d] & if_X = if_Y \arrow[dd] \\
R : X \frown^{\mathrm{U}^0}_0 Y \arrow[d] & \\
R : X \frown^{\mathrm{U}^0}_1 Y \arrow[r] \arrow[d] & if_X \frown^{B\,R}_0 if_Y \arrow[d] \\
\top \arrow[r] & \top
\end{tikzcd}$$

con : types $\to$ types

$$B : \mathrm{U}^0 \to \mathrm{U}^0$$

$$B\,X = \mathrm{Bool} \to X \to X \to X$$

$$\begin{tikzcd}
X = Y \arrow[r] \arrow[d] & B\,X = B\,Y \arrow[d] \\
X \frown^{\mathrm{U}^0}_0 Y \arrow[r, dotted, no head] \arrow[d] & B\,X \frown^{\mathrm{U}^0}_0 B\,Y \arrow[d] \\
X \frown^{\mathrm{U}^0}_1 Y \arrow[r] \arrow[d] & B\,X \frown^{\mathrm{U}^0}_1 B\,Y \arrow[d] \\
\top \arrow[r] & \top
\end{tikzcd}$$

Understanding modalities: Parametricity

par : types $\to$ values

$$\mathit{if} : (\textbf{par} \mid X : \mathrm{U}^0) \to B\,X$$

$$\begin{tikzcd}
X = Y \arrow[r] \arrow[d] & \mathit{if}_X = \mathit{if}_Y \arrow[dd] \\
R : X \frown^{\mathrm{U}^0}_0 Y \arrow[d] & \\
R : X \frown^{\mathrm{U}^0}_1 Y \arrow[r] \arrow[d] & \mathit{if}_X \frown^{B\,R}_0 \mathit{if}_Y \arrow[d] \\
\top \arrow[r] & \top
\end{tikzcd}$$

con : types $\to$ types

$$B : \mathrm{U}^0 \to \mathrm{U}^0$$

$$B\,X = \mathrm{Bool} \to X \to X \to X$$

$$\begin{tikzcd}
X = Y \arrow[r] \arrow[d] & B\,X = B\,Y \arrow[d] \\
X \frown^{\mathrm{U}^0}_0 Y \arrow[r] \arrow[d] & B\,X \frown^{\mathrm{U}^0}_0 B\,Y \arrow[d] \\
X \frown^{\mathrm{U}^0}_1 Y \arrow[r] \arrow[d] & B\,X \frown^{\mathrm{U}^0}_1 B\,Y \arrow[d] \\
\top \arrow[r] & \top
\end{tikzcd}$$

Understanding modalities: Irrelevance

irr : values $\to$ values

$$[\,] : (\mathbf{irr} \mid n : \mathbb{N}) \to \mathrm{List}_n\, A$$

$$\begin{tikzcd}
m = n \arrow[r] \arrow[d] & {[\,]_m = [\,]_n} \arrow[d] \\
r : m \frown^{\mathbb{N}}_0 n \arrow[d] & {[\,]_m \frown^{\mathrm{List}_\bullet\, A}_0 [\,]_n} \arrow[d] \\
\bullet : \top \arrow[ur] & \top
\end{tikzcd}$$

shi : values $\to$ types

$$\lambda n.\mathrm{List}_n\, A : (\mathbf{shi} \mid n : \mathbb{N}) \to \mathrm{U}^0$$

$$\begin{tikzcd}
m = n \arrow[r] \arrow[dd] & \mathrm{List}_m\, A = \mathrm{List}_n\, A \arrow[d] \\
 & \mathrm{List}_m\, A \frown^{\mathrm{U}^0}_0 \mathrm{List}_n\, A \arrow[d] \\
m \frown^{\mathbb{N}}_0 n \arrow[d] \arrow[ur, dotted] & \mathrm{List}_m\, A \frown^{\mathrm{U}^0}_1 \mathrm{List}_n\, A \arrow[d] \\
\top \arrow[ur, dotted] & \top
\end{tikzcd}$$

Understanding modalities: Irrelevance

irr : values → values

$$[\,] : (\mathbf{irr} \mid n : \mathbb{N}) \to \mathsf{List}_n\, A$$

$$\begin{tikzcd}
m = n \arrow[rr] \arrow[d] & & {[\,]_m = [\,]_n} \arrow[d] \\
r : m \frown^{\mathbb{N}}_{0} n \arrow[d] & & {[\,]_m \frown^{\mathsf{List}_\bullet\, A}_{0} [\,]_n} \arrow[d] \\
\bullet : \top \arrow[urr] & & \top
\end{tikzcd}$$

shi : values → types

$$\lambda n.\mathsf{List}_n\, A : (\mathbf{shi} \mid n : \mathbb{N}) \to \mathsf{U}^0$$

$$\begin{tikzcd}
m = n \arrow[rr] \arrow[dd] & & \mathsf{List}_m\, A = \mathsf{List}_n\, A \arrow[d] \\
& & \mathsf{List}_m\, A \frown^{\mathsf{U}^0}_{0} \mathsf{List}_n\, A \arrow[d] \\
m \frown^{\mathbb{N}}_{0} n \arrow[d] \arrow[urr, dotted] & & \mathsf{List}_m\, A \frown^{\mathsf{U}^0}_{1} \mathsf{List}_n\, A \arrow[d] \\
\top \arrow[urr, dotted] & & \top
\end{tikzcd}$$

Understanding modalities: Irrelevance

irr : values → values

$$[\,] : (\mathbf{irr} \mid n : \mathbb{N}) \to \mathsf{List}_n\, A$$

$$\begin{tikzcd}
m = n \arrow[r] \arrow[dd] & {[\,]_m = [\,]_n} \arrow[dd] \\
& \\
r : m \frown^{\mathbb{N}}_{0} n \arrow[d] & {[\,]_m \frown^{\mathsf{List}_\bullet\, A}_{0} [\,]_n} \arrow[d] \\
\bullet : \top \arrow[ur] & \top
\end{tikzcd}$$

shi : values → types

$$\lambda n.\mathsf{List}_n\, A : (\mathbf{shi} \mid n : \mathbb{N}) \to \mathsf{U}^0$$

$$\begin{tikzcd}
m = n \arrow[r] \arrow[dd] & \mathsf{List}_m\, A = \mathsf{List}_n\, A \arrow[d] \\
& \mathsf{List}_m\, A \frown^{\mathsf{U}^0}_{0} \mathsf{List}_n\, A \arrow[d] \\
m \frown^{\mathbb{N}}_{0} n \arrow[d] \arrow[ur, dotted, no head] & \mathsf{List}_m\, A \frown^{\mathsf{U}^0}_{1} \mathsf{List}_n\, A \arrow[d] \\
\top \arrow[ur, dotted, no head] & \top
\end{tikzcd}$$

Understanding modalities: Irrelevance

irr : values → values

$$[\,] : (\mathbf{irr} \mid n : \mathbb{N}) \to \mathrm{List}_n\ A$$

$$\begin{tikzcd}
m = n \arrow[rr] \arrow[d] & & {[\,]_m = [\,]_n} \arrow[d] \\
r : m \frown^{\mathbb{N}}_0 n \arrow[d] & & {[\,]_m \frown^{\mathrm{List}_\bullet\ A}_0 [\,]_n} \arrow[d] \\
\bullet : \top \arrow[urr] & & \top
\end{tikzcd}$$

shi : values → types

$$\lambda n.\mathrm{List}_n\ A : (\mathbf{shi} \mid n : \mathbb{N}) \to \mathrm{U}^0$$

$$\begin{tikzcd}
m = n \arrow[r] \arrow[dd] & \mathrm{List}_m\ A = \mathrm{List}_n\ A \arrow[d] \\
& \mathrm{List}_m\ A \frown^{\mathrm{U}^0}_0 \mathrm{List}_n\ A \arrow[d] \\
m \frown^{\mathbb{N}}_0 n \arrow[d] \arrow[ur, dotted, no head] & \mathrm{List}_m\ A \frown^{\mathrm{U}^0}_1 \mathrm{List}_n\ A \arrow[d] \\
\top \arrow[ur, dotted, no head] & \top
\end{tikzcd}$$

Understanding modalities: Irrelevance

irr : values → values

$$[\,] : (\mathbf{irr} \mid n : \mathbb{N}) \to \mathrm{List}_n\ A$$

$$\begin{tikzcd}
m = n \arrow[rr] \arrow[dd] & & {[\,]_m = [\,]_n} \arrow[dd] \\
& & \\
r : m \overset{\mathbb{N}}{\underset{0}{\frown}} n \arrow[d] & & {[\,]_m \overset{\mathrm{List}_\bullet\ A}{\underset{0}{\frown}} [\,]_n} \arrow[d] \\
\bullet : \top \arrow[urr] & & \top
\end{tikzcd}$$

shi : values → types

$$\lambda n.\mathrm{List}_n\ A : (\mathbf{shi} \mid n : \mathbb{N}) \to \mathrm{U}^0$$

$$\begin{tikzcd}
m = n \arrow[r] \arrow[dd] & \mathrm{List}_m\ A = \mathrm{List}_n\ A \arrow[d] \\
& \mathrm{List}_m\ A \overset{\mathrm{U}^0}{\underset{0}{\frown}} \mathrm{List}_n\ A \arrow[d] \arrow[dl, dash, dotted] \\
m \overset{\mathbb{N}}{\underset{0}{\frown}} n \arrow[d] & \mathrm{List}_m\ A \overset{\mathrm{U}^0}{\underset{1}{\frown}} \mathrm{List}_n\ A \arrow[d] \\
\top \arrow[ur] & \top
\end{tikzcd}$$

Understanding modalities: Irrelevance

irr : values → values

$$[\,] : (\mathbf{irr} \mid n : \mathbb{N}) \to \mathrm{List}_n\ A$$

$$\begin{tikzcd}
m = n \arrow[r] \arrow[d] & {[\,]_m = [\,]_n} \arrow[d] \\
r : m \frown^{\mathbb{N}}_{0} n \arrow[d] & {[\,]_m \frown^{\mathrm{List}_\bullet\ A}_{0} [\,]_n} \arrow[d] \\
\bullet : \top \arrow[ur] & \top
\end{tikzcd}$$

shi : values → types

$$\lambda n.\mathrm{List}_n\ A : (\mathbf{shi} \mid n : \mathbb{N}) \to \mathrm{U}^0$$

$$\begin{tikzcd}
m = n \arrow[r] \arrow[dd] & \mathrm{List}_m\ A = \mathrm{List}_n\ A \arrow[d] \\
 & \mathrm{List}_m\ A \frown^{\mathrm{U}^0}_{0} \mathrm{List}_n\ A \arrow[d] \\
m \frown^{\mathbb{N}}_{0} n \arrow[ur] \arrow[d] & \mathrm{List}_m\ A \frown^{\mathrm{U}^0}_{1} \mathrm{List}_n\ A \arrow[d] \\
\top \arrow[ur] & \top
\end{tikzcd}$$

The mode theory DoR

RelDTT can be built on an instance of MTT (Multimode Type Theory) with mode theory DoR:

- Modes are **depths** $p \in \mathbb{Z}_{\geq -1}$
- Modalities $\mu : p \to q$ are

  functions $\{0 \leq \ldots \leq q\} \to \{(=) \leq 0 \leq \ldots \leq p \leq \top\} : i \mapsto i \cdot \mu$

  where $f : (\mu \mid x : A) \to B(x)$ sends

  $$(r : x \frown^{A}_{i \cdot \mu} y) \quad \to \quad f(x) \frown^{B(r)}_{i} f(y).$$

  Modal types:

  $$\mathrm{mod}_\mu\, x \frown^{\langle \mu \mid A \rangle}_{i} \mathrm{mod}_\mu\, y \quad = \quad x \frown^{A}_{i \cdot \mu} y$$

- 2-cells are degree-wise inequalities.

Depth p is modelled in cubical sets with $p + 1$ different dimension flavours.

The mode theory DoR

RelDTT can be built on an instance of MTT (Multimode Type Theory) with mode theory DoR:

- Modes are **depths** $p \in \mathbb{Z}_{\geq -1}$
- Modalities $\mu : p \to q$ are

  functions $\{0 \leq \ldots \leq q\} \to \{(=) \leq 0 \leq \ldots \leq p \leq \top\} : i \mapsto i \cdot \mu$

  where $f : (\mu \mid x : A) \to B(x)$ sends

  $$(r : x \frown^{A}_{i \cdot \mu} y) \quad \to \quad f(x) \frown^{B(r)}_{i} f(y).$$

  Modal types:

  $$\mathrm{mod}_{\mu}\, x \frown^{\langle \mu \mid A \rangle}_{i} \mathrm{mod}_{\mu}\, y \quad = \quad x \frown^{A}_{i \cdot \mu} y$$

- 2-cells are degree-wise inequalities.

Depth p is modelled in cubical sets with $p+1$ different dimension flavours.

The mode theory DoR

RelDTT can be built on an instance of MTT (Multimode Type Theory) with mode theory DoR:

- Modes are **depths** $p \in \mathbb{Z}_{\geq -1}$
- Modalities $\mu : p \to q$ are

  functions $\{0 \leq \ldots \leq q\} \to \{(=) \leq 0 \leq \ldots \leq p \leq \top\} : i \mapsto i \cdot \mu$

  where $f : (\mu \mid x : A) \to B(x)$ sends

$$(r : x \frown^{A}_{i \cdot \mu} y) \quad \to \quad f(x) \frown^{B(r)}_{i} f(y).$$

  Modal types:

$$\mathrm{mod}_{\mu}\, x \frown^{\langle \mu | A \rangle}_{i} \mathrm{mod}_{\mu}\, y \quad = \quad x \frown^{A}_{i \cdot \mu} y$$

- 2-cells are degree-wise inequalities.

Depth p is modelled in cubical sets with $p + 1$ different dimension flavours.

The mode theory DoR

RelDTT can be built on an instance of MTT (Multimode Type Theory) with mode theory DoR:

- Modes are **depths** $p \in \mathbb{Z}_{\geq -1}$
- Modalities $\mu : p \to q$ are

  functions $\{0 \leq \ldots \leq q\} \to \{(=) \leq 0 \leq \ldots \leq p \leq \top\} : i \mapsto i \cdot \mu$

  where $f : (\mu \mid x : A) \to B(x)$ sends

$$(r : x \frown^{A}_{i \cdot \mu} y) \quad \to \quad f(x) \frown^{B(r)}_{i} f(y).$$

  Modal types:

$$\mathrm{mod}_\mu\, x \frown^{\langle \mu \mid A \rangle}_{i} \mathrm{mod}_\mu\, y \quad = \quad x \frown^{A}_{i \cdot \mu} y$$

- 2-cells are degree-wise inequalities.

Depth p is modelled in cubical sets with $p + 1$ different dimension flavours.

The mode theory DoR

RelDTT can be built on an instance of MTT (Multimode Type Theory) with mode theory DoR:

- Modes are **depths** $p \in \mathbb{Z}_{\geq -1}$
- Modalities $\mu : p \to q$ are

  functions $\{0 \leq \ldots \leq q\} \to \{(=) \leq 0 \leq \ldots \leq p \leq \top\} : i \mapsto i \cdot \mu$

  where $f : (\mu \mid x : A) \to B(x)$ sends

  $$(r : x \frown^{A}_{i \cdot \mu} y) \quad \to \quad f(x) \frown^{B(r)}_{i} f(y).$$

  Modal types:

  $$\mathrm{mod}_\mu\, x \frown^{\langle \mu | A \rangle}_{i} \mathrm{mod}_\mu\, y \quad = \quad x \frown^{A}_{i \cdot \mu} y$$

- 2-cells are degree-wise inequalities.

Depth p is modelled in cubical sets with $p + 1$ different dimension flavours.

The mode theory DoR

RelDTT can be built on an instance of MTT (Multimode Type Theory) with mode theory DoR:

- Modes are **depths** $p \in \mathbb{Z}_{\geq -1}$
- Modalities $\mu : p \to q$ are

  functions $\{0 \leq \ldots \leq q\} \to \{(=) \leq 0 \leq \ldots \leq p \leq \top\} : i \mapsto i \cdot \mu$

  where $f : (\mu \mid x : A) \to B(x)$ sends

  $$(r : x \frown^{A}_{i \cdot \mu} y) \quad \to \quad f(x) \frown^{B(r)}_{i} f(y).$$

  Modal types:

  $$\mathrm{mod}_\mu\, x \frown^{\langle \mu | A \rangle}_{i} \mathrm{mod}_\mu\, y \quad = \quad x \frown^{A}_{i \cdot \mu} y$$

- 2-cells are degree-wise inequalities.

Depth p is modelled in cubical sets with $p+1$ different dimension flavours.

Three Approaches to the Model

$$
\begin{tikzcd}[column sep=huge, row sep=huge]
\begin{array}{c}\text{Tamsamani \& Simpson's}\\ \text{model of 2-categories}\\ \text{(2-simplicial sets)}\end{array} \arrow[r, "\text{heterogenize}"'] \arrow[d, "\begin{array}{c}\text{higher}\\ \text{dimension}\end{array}"] & \begin{array}{c}\text{Pro-arrow}\\ \text{(pre-)equipments}\\ \text{[Woo82,Woo85]}\end{array} \arrow[d, "\begin{array}{c}\text{higher}\\ \text{dimension}\end{array}"'] & \begin{array}{c}\text{Bridge/path}\\ \text{cubical sets}\\ \text{(ParamDTT)}\\ \text{[NVD17]}\end{array} \arrow[l, "\text{directify}"] \arrow[d, "\begin{array}{c}\text{higher}\\ \text{dimension}\end{array}"'] \\
\begin{array}{c}\text{Tamsamani \& Simpson's}\\ \text{model of } n\text{-categories}\\ (n\text{-simplicial sets})\\ \text{[CL04,Tam99,Sim97]}\end{array} \arrow[r, "\text{heterogenize}"'] & \begin{array}{c}\textbf{Naturality}\\ \textbf{(Pre)type Theory}\end{array} & \begin{array}{c}\text{Degrees of}\\ \text{Relatedness}\\ \text{(RelDTT)}\\ \text{[ND18]}\end{array} \arrow[l, "\text{directify}"]
\end{tikzcd}
$$

Higher Pro-arrows:
Directifying
Degrees of Relatedness

- Equip types with **multiple, proof-relevant relations** $s \multimap_i t$ indexed by **degree** i
  - Proofs called ***i*-jets** (pro^{i-1}-arrows).
- Describe **function behaviour** by saying how functions **influence degree** and **orientation** of jets.

Degrees of Relatedness: **Four Laws**

- **Reflexivity:** $(a = b : A) \to ((a : A) \frown_i^A (b : A))$
  (Semantically, prop. eq. = def. eq.)
- **Degradation:** $((a : A) \frown_i^R (b : B)) \to ((a : A) \frown_{i+1}^R (b : B))$
- **Dependency:** $(a : A) \frown_i^R (b : B)$ **presumes** $R : A \frown_{i+1}^{\mathsf{U}} B$
- **Identity extension:** $(a : A) \frown_0^A (b : A)$ means $a = b : A$.
  $\rightsquigarrow$ heterogeneous $\frown_0$ serves as heterogeneous equality.

Pretypes!

Higher equipments: **Three Laws**

- **Reflexivity:** $(a = b : A) \to ((a : A) \twoheadrightarrow_i^A (b : A))$
- **Companion** $\varphi^\ddagger$ / **conjoint** $\varphi^\dagger$**:** $((a : A) \to_i^J (b : B)) \to ((a : A) \leftrightarrow_{i+1}^J (b : B))$
- **Dependency:** $(a : A) \twoheadrightarrow_i^J (b : B)$ **presumes** $J : A \twoheadrightarrow_{i+1}^{\mathsf{U}} B$

Degrees of Relatedness: **Four Laws**

- **Reflexivity:** $(a = b : A) \to ((a : A) \frown_i^A (b : A))$
  (Semantically, prop. eq. = def. eq.)
- **Degradation:** $((a : A) \frown_i^R (b : B)) \to ((a : A) \frown_{i+1}^R (b : B))$
- **Dependency:** $(a : A) \frown_i^R (b : B)$ **presumes** $R : A \frown_{i+1}^{\mathsf{U}} B$
- **Identity extension:** $(a : A) \frown_0^A (b : A)$ means $a = b : A$.
  $\rightsquigarrow$ heterogeneous $\frown_0$ serves as heterogeneous equality.

Pretypes!

Higher equipments: **Three Laws**

- **Reflexivity:** $(a = b : A) \to ((a : A) \multimap_i^A (b : A))$
- **Companion** $\varphi^\ddagger$ / **conjoint** $\varphi^\dagger$**:** $((a : A) \multimap_i^J (b : B)) \to ((a : A) \leftarrow\!\!\!\multimap_{i+1}^J (b : B))$
- **Dependency:** $(a : A) \multimap_i^J (b : B)$ **presumes** $J : A \multimap_{i+1}^{\mathsf{U}} B$

Degrees of Relatedness: **Four Laws**

- **Reflexivity:** $(a = b : A) \to ((a : A) \frown^{A}_{i} (b : A))$
  (Semantically, prop. eq. = def. eq.)
- **Degradation:** $((a : A) \frown^{R}_{i} (b : B)) \to ((a : A) \frown^{R}_{i+1} (b : B))$
- **Dependency:** $(a : A) \frown^{R}_{i} (b : B)$ **presumes** $R : A \frown^{\mathsf{U}}_{i+1} B$
- **Identity extension:** $(a : A) \frown^{A}_{0} (b : A)$ means $a = b : A$.
  $\rightsquigarrow$ heterogeneous $\frown_0$ serves as heterogeneous equality.

Pretypes!

Higher equipments: **Three Laws**

- **Reflexivity:** $(a = b : A) \to ((a : A) \mathrel{\rightarrow\!\!\triangleright}^{A}_{i} (b : A))$
- **Companion** $\varphi^{\ddagger}$ / **conjoint** $\varphi^{\dagger}$**:** $((a : A) \mathrel{\rightarrow\!\!\triangleright}^{J}_{i} (b : B)) \to ((a : A) \mathrel{\triangleleft\!\!\leftrightarrow\!\!\triangleright}^{J}_{i+1} (b : B))$
- **Dependency:** $(a : A) \mathrel{\rightarrow\!\!\triangleright}^{J}_{i} (b : B)$ **presumes** $J : A \mathrel{\rightarrow\!\!\triangleright}^{\mathsf{U}}_{i+1} B$

Degrees of Relatedness: **Four Laws**

- **Reflexivity:** $(a = b : A) \to ((a : A) \frown_i^A (b : A))$
  (Semantically, prop. eq. = def. eq.)
- **Degradation:** $((a : A) \frown_i^R (b : B)) \to ((a : A) \frown_{i+1}^R (b : B))$
- **Dependency:** $(a : A) \frown_i^R (b : B)$ **presumes** $R : A \frown_{i+1}^{\mathsf{U}} B$
- **Identity extension:** $(a : A) \frown_0^A (b : A)$ means $a = b : A$.
  $\rightsquigarrow$ heterogeneous $\frown_0$ serves as heterogeneous equality.

Pretypes!

Higher equipments: **Three Laws**

- **Reflexivity:** $(a = b : A) \to ((a : A) \rightarrow_i^A (b : A))$
- **Companion** $\varphi^\ddagger$ / **conjoint** $\varphi^\dagger$**:** $((a : A) \rightarrow_i^J (b : B)) \to ((a : A) \leftrightarrow_{i+1}^J (b : B))$
- **Dependency:** $(a : A) \rightarrow_i^J (b : B)$ **presumes** $J : A \rightarrow_{i+1}^{\mathsf{U}} B$

Degrees of Relatedness: **Four Laws**

- **Reflexivity:** $(a = b : A) \to ((a : A) \frown_i^A (b : A))$
  (Semantically, prop. eq. = def. eq.)
- **Degradation:** $((a : A) \frown_i^R (b : B)) \to ((a : A) \frown_{i+1}^R (b : B))$
- **Dependency:** $(a : A) \frown_i^R (b : B)$ **presumes** $R : A \frown_{i+1}^{\mathrm{U}} B$
- **Identity extension:** $(a : A) \frown_0^A (b : A)$ means $a = b : A$.
  $\rightsquigarrow$ heterogeneous $\frown_0$ serves as heterogeneous equality.

Pretypes!

Higher equipments: **Three Laws**

- **Reflexivity:** $(a = b : A) \to ((a : A) \multimap_i^A (b : A))$
- **Companion $\varphi^{\ddagger}$ / conjoint $\varphi^{\dagger}$:** $((a : A) \multimap_i^J (b : B)) \to ((a : A) \leftrightarrow_{i+1}^J (b : B))$
- **Dependency:** $(a : A) \multimap_i^J (b : B)$ **presumes** $J : A \multimap_{i+1}^{\mathrm{U}} B$

Understanding degrees

$(a : A) \rightharpoonup_0^f (b : B)$	a maps to b along ...
$f : (A : \mathrm{U}^0) \rightharpoonup_1^{\mathrm{U}^0} (B : \mathrm{U}^0)$	Any function f.
$\varphi : (G : \mathrm{Grp}) \rightharpoonup_1^{\mathrm{Grp}} (H : \mathrm{Grp})$	Any morphism φ.
$\psi : (G : \mathrm{Grp}) \rightharpoonup_1^{\mathscr{P}} (M : \mathrm{Monoid})$	Any heterogeneous morphism ψ along ...
$\mathscr{P} : (\mathrm{Grp} : \mathrm{U}^1) \rightharpoonup_2^{\mathrm{U}^1} (\mathrm{Monoid} : \mathrm{U}^1)$	Any profunctor $\mathscr{P}$ e.g. $\mathscr{P} = \mathrm{Hom}_{\mathrm{Monoid}}(U_{\mathrm{Grp}} \sqcup, \sqcup)$

Understanding degrees

$(a : A) \rightharpoondown_0^{f} (b : B)$ — *a* maps to *b* along ...

$f : (A : \mathrm{U}^0) \rightharpoondown_1^{\mathrm{U}^0} (B : \mathrm{U}^0)$ — Any function f.

$\varphi : (G : \mathrm{Grp}) \rightharpoondown_1^{\mathrm{Grp}} (H : \mathrm{Grp})$ — Any morphism φ.

$\psi : (G : \mathrm{Grp}) \rightharpoondown_1^{\mathscr{P}} (M : \mathrm{Monoid})$ — Any heterogeneous morphism ψ along ...

$\mathscr{P} : (\mathrm{Grp} : \mathrm{U}^1) \rightharpoondown_2^{\mathrm{U}^1} (\mathrm{Monoid} : \mathrm{U}^1)$ — Any profunctor $\mathscr{P}$

e.g. $\mathscr{P} = \mathrm{Hom}_{\mathrm{Monoid}}(U_{\mathrm{Grp}} \sqcup, \sqcup)$

Understanding degrees

$(a : A) \to_0^{f} (b : B)$ — a maps to b along ...

$f : (A : \mathrm{U}^0) \to_1^{\mathrm{U}^0} (B : \mathrm{U}^0)$ — Any function f.

$\varphi : (G : \mathrm{Grp}) \to_1^{\mathrm{Grp}} (H : \mathrm{Grp})$ — Any morphism φ.

$\psi : (G : \mathrm{Grp}) \to_1^{\mathscr{P}} (M : \mathrm{Monoid})$ — Any heterogeneous morphism ψ along ...

$\mathscr{P} : (\mathrm{Grp} : \mathrm{U}^1) \to_2^{\mathrm{U}^1} (\mathrm{Monoid} : \mathrm{U}^1)$ — Any profunctor $\mathscr{P}$

e.g. $\mathscr{P} = \mathrm{Hom}_{\mathrm{Monoid}}(U_{\mathrm{Grp}} \sqcup, \sqcup)$

Understanding degrees

$(a : A) \multimap^{f}_{0} (b : B)$	a maps to b along ...
$f : (A : \mathrm{U}^0) \multimap^{\mathrm{U}^0}_{1} (B : \mathrm{U}^0)$	Any function f.
$\varphi : (G : \mathrm{Grp}) \multimap^{\mathrm{Grp}}_{1} (H : \mathrm{Grp})$	Any morphism φ.
$\psi : (G : \mathrm{Grp}) \multimap^{\mathscr{P}}_{1} (M : \mathrm{Monoid})$	Any heterogeneous morphism ψ along ...
$\mathscr{P} : (\mathrm{Grp} : \mathrm{U}^1) \multimap^{\mathrm{U}^1}_{2} (\mathrm{Monoid} : \mathrm{U}^1)$	Any profunctor $\mathscr{P}$ e.g. $\mathscr{P} = \mathrm{Hom}_{\mathrm{Monoid}}(U_{\mathrm{Grp}}\sqcup, \sqcup)$

Understanding degrees

$(a : A) \rightharpoondown^{f}_{0} (b : B)$	a maps to b along ...
$f : (A : \mathrm{U}^0) \rightharpoondown^{\mathrm{U}^0}_{1} (B : \mathrm{U}^0)$	Any function f.
$\varphi : (G : \mathrm{Grp}) \rightharpoondown^{\mathrm{Grp}}_{1} (H : \mathrm{Grp})$	Any morphism φ.
$\psi : (G : \mathrm{Grp}) \rightharpoondown^{\mathscr{P}}_{1} (M : \mathrm{Monoid})$	Any heterogeneous morphism ψ along ...
$\mathscr{P} : (\mathrm{Grp} : \mathrm{U}^1) \rightharpoondown^{\mathrm{U}^1}_{2} (\mathrm{Monoid} : \mathrm{U}^1)$	Any profunctor $\mathscr{P}$
	e.g. $\mathscr{P} = \mathrm{Hom}_{\mathrm{Monoid}}(U_{\mathrm{Grp}} \sqcup, \sqcup)$

Understanding degrees

$(a : A) \rightarrowtriangle^{f}_{0} (b : B)$	a maps to b along ...
$f : (A : \mathrm{U}^0) \rightarrowtriangle^{\mathrm{U}^0}_{1} (B : \mathrm{U}^0)$	Any function f.
$\varphi : (G : \mathrm{Grp}) \rightarrowtriangle^{\mathrm{Grp}}_{1} (H : \mathrm{Grp})$	Any morphism φ.
$\psi : (G : \mathrm{Grp}) \rightarrowtriangle^{\mathscr{P}}_{1} (M : \mathrm{Monoid})$	Any heterogeneous morphism ψ along ...
$\mathscr{P} : (\mathrm{Grp} : \mathrm{U}^1) \rightarrowtriangle^{\mathrm{U}^1}_{2} (\mathrm{Monoid} : \mathrm{U}^1)$	Any profunctor $\mathscr{P}$ e.g. $\mathscr{P} = \mathrm{Hom}_{\mathrm{Monoid}}(U_{\mathrm{Grp}} \sqcup, \sqcup)$

Understanding modalities: Limits

$\mathbf{lim}^{\oplus}$

$$e : (\mathbf{lim}^{\oplus} \mid X : \mathrm{Grp}) \to |X|$$

$$\begin{tikzcd}
G = H \arrow[r] \arrow[d] & e_G = e_H \arrow[dd] \\
p : G \xrightarrow[0]{\mathrm{id}_{\mathrm{Grp}}} H \arrow[d, "\ddagger"'] & \\
\varphi : G \xrightarrow[1]{\mathrm{Hom}_{\mathrm{Grp}}} H \arrow[r] \arrow[d] & e_G \xrightarrow[0]{|\varphi|} e_H \arrow[d] \\
\top \arrow[r] & \top
\end{tikzcd}$$

$\mathbf{ftr}^{\oplus}$

$$|\sqcup| : (\mathbf{ftr}^{\oplus} \mid \mathrm{Grp}) \to \mathrm{U}^0$$

$$\begin{tikzcd}
G = H \arrow[r] \arrow[d] & |G| = |H| \arrow[d] \\
G \xrightarrow[0]{\mathrm{id}_{\mathrm{Grp}}} H \arrow[r, dotted, no head] \arrow[d, "\ddagger"'] & |G| \xrightarrow[0]{\mathrm{id}_{\mathrm{U}^0}} |H| \arrow[d, "\ddagger"] \\
G \xrightarrow[1]{\mathrm{Hom}_{\mathrm{Grp}}} H \arrow[r, dotted, no head] \arrow[d] & |G| \xrightarrow[1]{\mathrm{Hom}_{\mathrm{U}^0}} |H| \arrow[d] \\
\top \arrow[r] & \top
\end{tikzcd}$$

Understanding modalities: Limits

$\mathbf{lim}^{\oplus}$

$$e : (\mathbf{lim}^{\oplus} \shortmid X : \mathrm{Grp}) \to |X|$$

$$\begin{tikzcd}
G = H \arrow[rr] \arrow[d] & & e_G = e_H \arrow[dd] \\
p : G \rightarrowtriangle_0^{\mathrm{id}_{\mathrm{Grp}}} H \arrow[d, "\ddagger"'] & & \\
\varphi : G \rightarrowtriangle_1^{\mathrm{Hom}_{\mathrm{Grp}}} H \arrow[rr] \arrow[d] & & e_G \rightarrowtriangle_0^{|\varphi|} e_H \arrow[d] \\
\top \arrow[rr] & & \top
\end{tikzcd}$$

$\mathbf{ftr}^{\oplus}$

$$|\sqcup| : (\mathbf{ftr}^{\oplus} \shortmid \mathrm{Grp}) \to \mathrm{U}^0$$

$$\begin{tikzcd}
G = H \arrow[r] \arrow[d] & {|G| = |H|} \arrow[d] \\
G \rightarrowtriangle_0^{\mathrm{id}_{\mathrm{Grp}}} H \arrow[r, dotted, no head] \arrow[d, "\ddagger"'] & {|G| \rightarrowtriangle_0^{\mathrm{id}_{\mathrm{U}^0}} |H|} \arrow[d, "\ddagger"] \\
G \rightarrowtriangle_1^{\mathrm{Hom}_{\mathrm{Grp}}} H \arrow[r, dotted, no head] \arrow[d] & {|G| \rightarrowtriangle_1^{\mathrm{Hom}_{\mathrm{U}^0}} |H|} \arrow[d] \\
\top \arrow[r] & \top
\end{tikzcd}$$

Understanding modalities: Limits

$\mathbf{lim}^{\oplus}$

$$e : (\mathbf{lim}^{\oplus} \mid X : \mathrm{Grp}) \to |X|$$

$$\begin{tikzcd}
G = H \arrow[rr] \arrow[d] & & e_G = e_H \arrow[dd] \\
p : G \overset{\mathrm{id}_{\mathrm{Grp}}}{\rightarrowtriangle_0} H \arrow[d, "\ddagger"'] & & \\
\varphi : G \overset{\mathrm{Hom}_{\mathrm{Grp}}}{\rightarrowtriangle_1} H \arrow[rr] \arrow[d] & & e_G \overset{|\varphi|}{\rightarrowtriangle_0} e_H \arrow[d] \\
\top \arrow[rr] & & \top
\end{tikzcd}$$

$\mathbf{ftr}^{\oplus}$

$$|\sqcup| : (\mathbf{ftr}^{\oplus} \mid \mathrm{Grp}) \to \mathrm{U}^0$$

$$\begin{tikzcd}
G = H \arrow[rr] \arrow[d] & & {|G| = |H|} \arrow[d] \\
G \overset{\mathrm{id}_{\mathrm{Grp}}}{\rightarrowtriangle_0} H \arrow[rr, dotted, no head] \arrow[d, "\ddagger"'] & & {|G| \overset{\mathrm{id}_{\mathrm{U}^0}}{\rightarrowtriangle_0} |H|} \arrow[d, "\ddagger"] \\
G \overset{\mathrm{Hom}_{\mathrm{Grp}}}{\rightarrowtriangle_1} H \arrow[rr, dotted, no head] \arrow[d] & & {|G| \overset{\mathrm{Hom}_{\mathrm{U}^0}}{\rightarrowtriangle_1} |H|} \arrow[d] \\
\top \arrow[rr] & & \top
\end{tikzcd}$$

$\mathbf{lim}^{\oplus}$

$$e : (\mathbf{lim}^{\oplus} \shortmid X : \mathrm{Grp}) \to |X|$$

$$\begin{tikzcd}
G = H \arrow[rr] \arrow[d] & & e_G = e_H \arrow[dd] \\
p : G \rightarrow\!\!\!\triangleright^{\mathrm{id}_{\mathrm{Grp}}}_{0} H \arrow[d, "\ddagger"'] & & \\
\varphi : G \rightarrow\!\!\!\triangleright^{\mathrm{Hom}_{\mathrm{Grp}}}_{1} H \arrow[rr] \arrow[d] & & e_G \rightarrow\!\!\!\triangleright^{|\varphi|}_{0} e_H \arrow[d] \\
\top \arrow[rr] & & \top
\end{tikzcd}$$

$\mathbf{ftr}^{\oplus}$

$$|\sqcup| : (\mathbf{ftr}^{\oplus} \shortmid \mathrm{Grp}) \to \mathrm{U}^0$$

$$\begin{tikzcd}
G = H \arrow[rr] \arrow[d] & & {|G| = |H|} \arrow[d] \\
G \rightarrow\!\!\!\triangleright^{\mathrm{id}_{\mathrm{Grp}}}_{0} H \arrow[rr, dotted, no head] \arrow[d, "\ddagger"'] & & {|G| \rightarrow\!\!\!\triangleright^{\mathrm{id}_{\mathrm{U}^0}}_{0} |H|} \arrow[d, "\ddagger"] \\
G \rightarrow\!\!\!\triangleright^{\mathrm{Hom}_{\mathrm{Grp}}}_{1} H \arrow[rr] \arrow[d] & & {|G| \rightarrow\!\!\!\triangleright^{\mathrm{Hom}_{\mathrm{U}^0}}_{1} |H|} \arrow[d] \\
\top \arrow[rr] & & \top
\end{tikzcd}$$

Understanding modalities: Limits

$\mathbf{lim}^{\oplus}$

$$e : (\mathbf{lim}^{\oplus} \mid X : \mathsf{Grp}) \to |X|$$

$$\begin{tikzcd}
G = H \arrow[r] \arrow[d] & e_G = e_H \arrow[dd] \\
p : G \overset{\mathrm{id}_{\mathsf{Grp}}}{\rightarrowtriangle_0} H \arrow[d, "\ddagger"'] & \\
\varphi : G \overset{\mathrm{Hom}_{\mathsf{Grp}}}{\rightarrowtriangle_1} H \arrow[r] \arrow[d] & e_G \overset{|\varphi|}{\rightarrowtriangle_0} e_H \arrow[d] \\
\top \arrow[r] & \top
\end{tikzcd}$$

$\mathbf{ftr}^{\oplus}$

$$|\sqcup| : (\mathbf{ftr}^{\oplus} \mid \mathsf{Grp}) \to \mathsf{U}^0$$

$$\begin{tikzcd}
G = H \arrow[r] \arrow[d] & |G| = |H| \arrow[d] \\
G \overset{\mathrm{id}_{\mathsf{Grp}}}{\rightarrowtriangle_0} H \arrow[r] \arrow[d, "\ddagger"'] & |G| \overset{\mathrm{id}_{\mathsf{U}^0}}{\rightarrowtriangle_0} |H| \arrow[d, "\ddagger"] \\
G \overset{\mathrm{Hom}_{\mathsf{Grp}}}{\rightarrowtriangle_1} H \arrow[r] \arrow[d] & |G| \overset{\mathrm{Hom}_{\mathsf{U}^0}}{\rightarrowtriangle_1} |H| \arrow[d] \\
\top \arrow[r] & \top
\end{tikzcd}$$

Understanding modalities: Limits

$\mathbf{lim}^{\ominus}$

$$\mathrm{hd} : (\mathbf{lim}^{\ominus} \mid X : \mathrm{Coalg}_{\mathbb{N}\times\sqcup}) \to |X| \to \mathbb{N}$$

$$\begin{tikzcd}
X = Y \arrow[rr] \arrow[d] & & hd_X = hd_Y \arrow[dd] \\
p : X \xrightarrow{\mathrm{id}_{\mathrm{Coalg}_{\mathbb{N}\times\sqcup}}}_0 Y \arrow[d, "\ddagger"'] & & \\
\varphi : X \xrightarrow{\mathrm{Hom}_{\mathrm{Coalg}_{\mathbb{N}\times\sqcup}}}_1 Y \arrow[r, "\ominus"] \arrow[d] & hd_X \xleftarrow{\sqcup\circ|\varphi|}_0 hd_Y \arrow[r, phantom] & \arrow[d] \\
\top \arrow[rr] & & \top
\end{tikzcd}$$

$\mathbf{ftr}^{\ominus}$

$$\lambda X.(|X| \to \mathbb{N}) : (\mathbf{ftr}^{\ominus} \mid \mathrm{Coalg}_{\mathbb{N}\times\sqcup}) \to \mathrm{U}^0$$

$$\begin{tikzcd}
X = Y \arrow[r] \arrow[d] & \mathbb{N}^{|X|} = \mathbb{N}^{|Y|} \arrow[d] \\
X \xrightarrow{\mathrm{id}_{\mathrm{Coalg}_{\mathbb{N}\times\sqcup}}}_0 Y \arrow[r, dotted, no head] \arrow[d, "\ddagger"'] & \mathbb{N}^{|X|} \xrightarrow{\mathrm{id}_{\mathrm{U}^0}}_0 \mathbb{N}^{|Y|} \arrow[d, "\dagger"] \\
G \xrightarrow{\mathrm{Hom}_{\mathrm{Grp}}}_1 H \arrow[r, dotted, no head] \arrow[d] & \mathbb{N}^{|X|} \xleftarrow{\mathrm{Hom}_{\mathrm{U}^0}}_1 \mathbb{N}^{|Y|} \arrow[d] \\
\top \arrow[r] & \top
\end{tikzcd}$$

Understanding modalities: Limits

$\mathbf{lim}^{\ominus}$

$$\mathsf{hd} : (\mathbf{lim}^{\ominus} \mid X : \mathsf{Coalg}_{\mathbb{N}\times\sqcup}) \to |X| \to \mathbb{N}$$

$$\begin{tikzcd}
X = Y \arrow[rr] \arrow[d] & & hd_X = hd_Y \arrow[dd] \\
p : X \xrightarrow[0]{\mathrm{id}_{\mathsf{Coalg}_{\mathbb{N}\times\sqcup}}} Y \arrow[d, "\ddagger"'] & & \\
\varphi : X \xrightarrow[1]{\mathrm{Hom}_{\mathsf{Coalg}_{\mathbb{N}\times\sqcup}}} Y \arrow[r, "\ominus"] \arrow[d] & hd_X \xleftarrow[0]{\sqcup\circ|\varphi|} hd_Y \arrow[d] & \\
\top \arrow[r] & \top &
\end{tikzcd}$$

$\mathbf{ftr}^{\ominus}$

$$\lambda X.(|X| \to \mathbb{N}) : (\mathbf{ftr}^{\ominus} \mid \mathsf{Coalg}_{\mathbb{N}\times\sqcup}) \to \mathsf{U}^0$$

$$\begin{tikzcd}
X = Y \arrow[r] \arrow[d] & \mathbb{N}^{|X|} = \mathbb{N}^{|Y|} \arrow[d] \\
X \xrightarrow[0]{\mathrm{id}_{\mathsf{Coalg}_{\mathbb{N}\times\sqcup}}} Y \arrow[r, dotted, no head] \arrow[d, "\ddagger"'] & \mathbb{N}^{|X|} \xrightarrow[0]{\mathrm{id}_{\mathsf{U}^0}} \mathbb{N}^{|Y|} \arrow[d, "\dagger"] \\
G \xrightarrow[1]{\mathrm{Hom}_{\mathsf{Grp}}} H \arrow[r, dotted, no head] \arrow[d] & \mathbb{N}^{|X|} \xleftarrow[1]{\mathrm{Hom}_{\mathsf{U}^0}} \mathbb{N}^{|Y|} \arrow[d] \\
\top \arrow[r] & \top
\end{tikzcd}$$

Understanding modalities: Limits

$\mathbf{lim}^{\ominus}$

$$\mathsf{hd} : (\mathbf{lim}^{\ominus} \mid X : \mathsf{Coalg}_{\mathbb{N}\times\sqcup}) \to |X| \to \mathbb{N}$$

$$\begin{tikzcd}
X = Y \arrow[rr] \arrow[d] & & hd_X = hd_Y \arrow[dd] \\
p : X \xrightarrow{\mathrm{id}_{\mathsf{Coalg}_{\mathbb{N}\times\sqcup}}}_{0} Y \arrow[d, "\ddagger"'] & & \\
\varphi : X \xrightarrow{\mathrm{Hom}_{\mathsf{Coalg}_{\mathbb{N}\times\sqcup}}}_{1} Y \arrow[r, "\ominus"] \arrow[d] & hd_X \xleftarrow{\sqcup\circ|\varphi|}_{0} hd_Y \arrow[d] & \\
\top \arrow[r] & \top &
\end{tikzcd}$$

$\mathbf{ftr}^{\ominus}$

$$\lambda X.(|X| \to \mathbb{N}) : (\mathbf{ftr}^{\ominus} \mid \mathsf{Coalg}_{\mathbb{N}\times\sqcup}) \to \mathsf{U}^0$$

$$\begin{tikzcd}
X = Y \arrow[r] \arrow[d] & \mathbb{N}^{|X|} = \mathbb{N}^{|Y|} \arrow[d] \\
X \xrightarrow{\mathrm{id}_{\mathsf{Coalg}_{\mathbb{N}\times\sqcup}}}_{0} Y \arrow[r, dotted, no head] \arrow[d, "\ddagger"'] & \mathbb{N}^{|X|} \xrightarrow{\mathrm{id}_{\mathsf{U}^0}}_{0} \mathbb{N}^{|Y|} \arrow[d, "\dagger"] \\
G \xrightarrow{\mathrm{Hom}_{\mathsf{Grp}}}_{1} H \arrow[r, dotted, no head] \arrow[d] & \mathbb{N}^{|X|} \xleftarrow{\mathrm{Hom}_{\mathsf{U}^0}}_{1} \mathbb{N}^{|Y|} \arrow[d] \\
\top \arrow[r] & \top
\end{tikzcd}$$

Understanding modalities: Limits

$\mathbf{lim}^{\ominus}$

$$\mathsf{hd} : (\mathbf{lim}^{\ominus} \shortmid X : \mathsf{Coalg}_{\mathbb{N}\times\sqcup}) \to |X| \to \mathbb{N}$$

$$\begin{tikzcd}
X = Y \arrow[rr] \arrow[d] & & hd_X = hd_Y \arrow[dd] \\
p : X \mathrel{-\!\!\triangleright}_0^{\mathrm{id}_{\mathsf{Coalg}_{\mathbb{N}\times\sqcup}}} Y \arrow[d, "\ddagger"'] & & \\
\varphi : X \mathrel{-\!\!\triangleright}_1^{\mathrm{Hom}_{\mathsf{Coalg}_{\mathbb{N}\times\sqcup}}} Y \arrow[rr, "\ominus"] \arrow[d] & & hd_X \mathrel{\triangleleft\!\!-}_0^{\sqcup\circ|\varphi|} hd_Y \arrow[d] \\
\top \arrow[rr] & & \top
\end{tikzcd}$$

$\mathbf{ftr}^{\ominus}$

$$\lambda X.(|X| \to \mathbb{N}) : (\mathbf{ftr}^{\ominus} \shortmid \mathsf{Coalg}_{\mathbb{N}\times\sqcup}) \to \mathsf{U}^0$$

$$\begin{tikzcd}
X = Y \arrow[r] \arrow[d] & \mathbb{N}^{|X|} = \mathbb{N}^{|Y|} \arrow[d] \\
X \mathrel{-\!\!\triangleright}_0^{\mathrm{id}_{\mathsf{Coalg}_{\mathbb{N}\times\sqcup}}} Y \arrow[r, dotted, no head] \arrow[d, "\ddagger"'] & \mathbb{N}^{|X|} \mathrel{-\!\!\triangleright}_0^{\mathrm{id}_{\mathsf{U}^0}} \mathbb{N}^{|Y|} \arrow[d, "\dagger"] \\
G \mathrel{-\!\!\triangleright}_1^{\mathrm{Hom}_{\mathsf{Grp}}} H \arrow[r, "\ominus"] \arrow[d] & \mathbb{N}^{|X|} \mathrel{\triangleleft\!\!-}_1^{\mathrm{Hom}_{\mathsf{U}^0}} \mathbb{N}^{|Y|} \arrow[d] \\
\top \arrow[r] & \top
\end{tikzcd}$$

Understanding modalities: Limits

$\mathbf{lim}^{\ominus}$

$$\mathsf{hd} : (\mathbf{lim}^{\ominus} \shortmid X : \mathsf{Coalg}_{\mathbb{N}\times\sqcup}) \to |X| \to \mathbb{N}$$

$$\begin{tikzcd}
X = Y \arrow[rr] \arrow[d] & & hd_X = hd_Y \arrow[dd] \\
p : X \xrightarrow{\mathrm{id}_{\mathsf{Coalg}_{\mathbb{N}\times\sqcup}}}_0 Y \arrow[d, "\ddagger"'] & & \\
\varphi : X \xrightarrow{\mathrm{Hom}_{\mathsf{Coalg}_{\mathbb{N}\times\sqcup}}}_1 Y \arrow[r, "\ominus"] \arrow[d] & hd_X \xleftarrow{\sqcup\circ|\varphi|}_0 hd_Y \arrow[d] & \\
\top \arrow[r] & \top &
\end{tikzcd}$$

$\mathbf{ftr}^{\ominus}$

$$\lambda X.(|X| \to \mathbb{N}) : (\mathbf{ftr}^{\ominus} \shortmid \mathsf{Coalg}_{\mathbb{N}\times\sqcup}) \to \mathsf{U}^0$$

$$\begin{tikzcd}
X = Y \arrow[r] \arrow[d] & \mathbb{N}^{|X|} = \mathbb{N}^{|Y|} \arrow[d] \\
X \xrightarrow{\mathrm{id}_{\mathsf{Coalg}_{\mathbb{N}\times\sqcup}}}_0 Y \arrow[r] \arrow[d, "\ddagger"'] & \mathbb{N}^{|X|} \xrightarrow{\mathrm{id}_{\mathsf{U}^0}}_0 \mathbb{N}^{|Y|} \arrow[d, "\dagger"] \\
G \xrightarrow{\mathrm{Hom}_{\mathsf{Grp}}}_1 H \arrow[r, "\ominus"] \arrow[d] & \mathbb{N}^{|X|} \xleftarrow{}_1 \mathrm{Hom}_{\mathsf{U}^0}\ \mathbb{N}^{|Y|} \arrow[d] \\
\top \arrow[r] & \top
\end{tikzcd}$$

The mode theory

NatPT instantiates MTT (Multimode Type Theory) with:

- Modes are **dimensions** $p \in \mathbb{N}$ (+ you can mark a degree $i < n$ as symmetric)
- Modalities $\mu : p \to q$ are certain functions

  $\{0, \ldots, q-1\} \to \{(=), 0, \ldots, p-1, \top\} \times \{\circledast, \oplus, \ominus, \otimes\}$

  where $f : (\mu \mid x : A) \to B(x)$ sends

  $$(r : x \rightarrowtriangle^{A}_{i \cdot \mu} y) \quad \to \quad f(x) \rightarrowtriangle^{B(r)}_{i} f(y).$$

  Modal types:

  $$\mathrm{mod}_\mu\, x \rightarrowtriangle^{\langle \mu \mid A \rangle}_{i} \mathrm{mod}_\mu\, y \quad = \quad x \rightarrowtriangle^{A}_{i \cdot \mu} y$$

- 2-cells are degree-wise inequalities.

The mode theory

NatPT instantiates MTT (Multimode Type Theory) with:

- Modes are **dimensions** $p \in \mathbb{N}$ (+ you can mark a degree $i < n$ as symmetric)
- Modalities $\mu : p \to q$ are certain functions
  $\{0, \ldots, q-1\} \to \{(=), 0, \ldots, p-1, \top\} \times \{\circledast, \oplus, \ominus, \otimes\}$
  where $f : (\mu \mid x : A) \to B(x)$ sends

  $$(r : x \rightarrowtriangle^{A}_{i \cdot \mu} y) \quad \to \quad f(x) \rightarrowtriangle^{B(r)}_{i} f(y).$$

  Modal types:

  $$\mathrm{mod}_\mu\, x \rightarrowtriangle^{\langle \mu \mid A \rangle}_{i} \mathrm{mod}_\mu\, y \quad = \quad x \rightarrowtriangle^{A}_{i \cdot \mu} y$$

- 2-cells are degree-wise inequalities.

The mode theory

NatPT instantiates MTT (Multimode Type Theory) with:

- Modes are **dimensions** $p \in \mathbb{N}$ (+ you can mark a degree $i < n$ as symmetric)
- Modalities $\mu : p \to q$ are certain functions
  $\{0, \ldots, q-1\} \to \{(=), 0, \ldots, p-1, \top\} \times \{\circledast, \oplus, \ominus, \otimes\}$
  where $f : (\mu \mid x : A) \to B(x)$ sends

$$(r : x \rightharpoonup^{A}_{i \cdot \mu} y) \quad \to \quad f(x) \rightharpoonup^{B(r)}_{i} f(y).$$

Modal types:

$$\mathrm{mod}_\mu\, x \rightharpoonup^{\langle \mu | A \rangle}_{i} \mathrm{mod}_\mu\, y \quad = \quad x \rightharpoonup^{A}_{i \cdot \mu} y$$

- 2-cells are degree-wise inequalities.

The mode theory

NatPT instantiates MTT (Multimode Type Theory) with:

- Modes are **dimensions** $p \in \mathbb{N}$ (+ you can mark a degree $i < n$ as symmetric)
- Modalities $\mu : p \to q$ are certain functions
  $\{0, \ldots, q-1\} \to \{(=), 0, \ldots, p-1, \top\} \times \{\circledast, \oplus, \ominus, \otimes\}$
  where $f : (\mu \mid x : A) \to B(x)$ sends

$$(r : x \rightharpoondown^{A}_{i \cdot \mu} y) \quad \to \quad f(x) \rightharpoondown^{B(r)}_{i} f(y).$$

Modal types:

$$\mathrm{mod}_\mu\, x \rightharpoondown^{\langle \mu | A \rangle}_{i} \mathrm{mod}_\mu\, y \quad = \quad x \rightharpoondown^{A}_{i \cdot \mu} y$$

- 2-cells are degree-wise inequalities.

The mode theory

NatPT instantiates MTT (Multimode Type Theory) with:

- Modes are **dimensions** $p \in \mathbb{N}$ (+ you can mark a degree $i < n$ as symmetric)
- Modalities $\mu : p \to q$ are certain functions
  $\{0, \ldots, q-1\} \to \{(=), 0, \ldots, p-1, \top\} \times \{\circledast, \oplus, \ominus, \otimes\}$
  where $f : (\mu \mid x : A) \to B(x)$ sends

$$(r : x \rightharpoonup^{A}_{i \cdot \mu} y) \quad \to \quad f(x) \rightharpoonup^{B(r)}_{i} f(y).$$

Modal types:

$$\mathrm{mod}_{\mu}\, x \rightharpoonup^{\langle \mu | A \rangle}_{i} \mathrm{mod}_{\mu}\, y \quad = \quad x \rightharpoonup^{A}_{i \cdot \mu} y$$

- 2-cells are degree-wise inequalities.

Three Approaches to the Model

$$
\begin{tikzcd}[column sep=huge, row sep=huge]
\begin{array}{c}\text{Tamsamani \& Simpson's}\\ \text{model of 2-categories}\\ \text{(2-simplicial sets)}\end{array} \arrow[r, "\text{heterogenize}"'] \arrow[d, "\begin{array}{c}\text{higher}\\ \text{dimension}\end{array}"] & \begin{array}{c}\text{Pro-arrow}\\ \text{(pre-)equipments}\\ \text{[Woo82,Woo85]}\end{array} \arrow[d, "\begin{array}{c}\text{higher}\\ \text{dimension}\end{array}"'] & \begin{array}{c}\text{Bridge/path}\\ \text{cubical sets}\\ \text{(ParamDTT)}\\ \text{[NVD17]}\end{array} \arrow[l, "\text{directify}"] \arrow[d, "\begin{array}{c}\text{higher}\\ \text{dimension}\end{array}"'] \\
\begin{array}{c}\text{Tamsamani \& Simpson's}\\ \text{model of } n\text{-categories}\\ (n\text{-simplicial sets})\\ \text{[CL04,Tam99,Sim97]}\end{array} \arrow[r, "\text{heterogenize}"'] & \begin{array}{c}\textbf{Naturality}\\ \textbf{(Pre)type Theory}\end{array} & \begin{array}{c}\text{Degrees of}\\ \text{Relatedness}\\ \text{(RelDTT)}\\ \text{[ND18]}\end{array} \arrow[l, "\text{directify}"]
\end{tikzcd}
$$

The Model

The Twisted Prism Functor

Δ is a skeleton of (hence $\simeq$) NEFinLinOrd.

Twisted Prism Functor [PK20]

$\sqcup \ltimes \mathbb{I} : \mathrm{NEFinLinOrd} \to \mathrm{NEFinLinOrd} :$
$W \mapsto W^{\mathrm{op}} \uplus_{<} W$

$$a \longrightarrow b \qquad \mapsto \qquad \begin{tikzcd} (a,0) \arrow[d] & (b,0) \arrow[l] \arrow[d] \\ (a,1) \arrow[r] & (b,1) \end{tikzcd}$$

MTraS shape modelled by $\sqcup \ltimes \mathbb{I}$ reconciles:

- Hom as a **contra-/covariant bifunctor**,
- Hom as a **constrained function type**.

$\mathbb{I}$ as an MTraS-shape is better behaved on $\bowtie$:

Twisted Cube Category $\bowtie$ [PK20]

(Roughly) the subcategory of NEFinLinOrd (or Δ) generated by $\top$ and $\sqcup \ltimes \mathbb{I}$.

➔ Use $\bowtie$ instead of Δ.

ℹ Pinyo & Kraus carve $\bowtie$ out of **graph** category.

The Twisted Prism Functor

Δ is a skeleton of (hence $\simeq$) NEFinLinOrd.

Twisted Prism Functor [PK20]

$\sqcup \ltimes \mathbb{I} : \mathrm{NEFinLinOrd} \to \mathrm{NEFinLinOrd} :$
$W \mapsto W^{\mathrm{op}} \uplus_< W$

$$a \longrightarrow b \quad \mapsto \quad \begin{tikzcd} (a,0) \arrow[d] & (b,0) \arrow[l] \arrow[d] \\ (a,1) \arrow[r] & (b,1) \end{tikzcd}$$

MTraS shape modelled by $\sqcup \ltimes \mathbb{I}$ reconciles:

- Hom as a **contra-/covariant bifunctor**,
- Hom as a **constrained function type**.

$\mathbb{I}$ as an MTraS-shape is better behaved on $\bowtie$:

Twisted Cube Category $\bowtie$ [PK20]

(Roughly) the subcategory of NEFinLinOrd (or Δ) generated by $\top$ and $\sqcup \ltimes \mathbb{I}$.

➔ Use $\bowtie$ instead of Δ.

ℹ Pinyo & Kraus carve $\bowtie$ out of **graph** category.

The Twisted Prism Functor

Δ is a skeleton of (hence $\simeq$) NEFinLinOrd.

Twisted Prism Functor [PK20]

$\sqcup \ltimes \mathbb{I} : \mathrm{NEFinLinOrd} \to \mathrm{NEFinLinOrd} :$
$W \mapsto W^{\mathrm{op}} \uplus_{<} W$

$$a \longrightarrow b \qquad \mapsto \qquad \begin{tikzcd} (a,0) \arrow[d] & (b,0) \arrow[l] \arrow[d] \\ (a,1) \arrow[r] & (b,1) \end{tikzcd}$$

MTraS shape modelled by $\sqcup \ltimes \mathbb{I}$ reconciles:

- Hom as a **contra-/covariant bifunctor**,
- Hom as a **constrained function type**.

$\mathbb{I}$ as an MTraS-shape is better behaved on $\bowtie$:

Twisted Cube Category $\bowtie$ [PK20]

(Roughly) the subcategory of NEFinLinOrd (or Δ) generated by $\top$ and $\sqcup \ltimes \mathbb{I}$.

➔ Use $\bowtie$ instead of Δ.

ℹ Pinyo & Kraus carve $\bowtie$ out of **graph** category.

The Twisted Prism Functor

Δ is a skeleton of (hence $\simeq$) NEFinLinOrd.

Twisted Prism Functor [PK20]

$\sqcup \ltimes \mathbb{I}$: NEFinLinOrd $\to$ NEFinLinOrd :
$W \mapsto W^{\mathrm{op}} \uplus_< W$

$$a \longrightarrow b \qquad \mapsto \qquad \begin{tikzcd} (a,0) \arrow[d] & (b,0) \arrow[l] \arrow[d] \\ (a,1) \arrow[r] & (b,1) \end{tikzcd}$$

MTraS shape modelled by $\sqcup \ltimes \mathbb{I}$ reconciles:

- Hom as a **contra-/covariant bifunctor**,
- Hom as a **constrained function type**.

$\mathbb{I}$ as an MTraS-shape is better behaved on $\bowtie$:

Twisted Cube Category $\bowtie$ [PK20]

(Roughly) the subcategory of NEFinLinOrd (or Δ) generated by $\top$ and $\sqcup \ltimes \mathbb{I}$.

➔ Use $\bowtie$ instead of Δ.

ℹ Pinyo & Kraus carve $\bowtie$ out of **graph** category.

The Twisted Prism Functor

Δ is a skeleton of (hence $\simeq$) NEFinLinOrd.

Twisted Prism Functor [PK20]

$\sqcup \ltimes \mathbb{I}$: NEFinLinOrd $\to$ NEFinLinOrd :
$W \mapsto W^{\mathrm{op}} \uplus_< W$

$$a \longrightarrow b \qquad \mapsto \qquad \begin{tikzcd} (a,0) \arrow[d] & (b,0) \arrow[l] \arrow[d] \\ (a,1) \arrow[r] & (b,1) \end{tikzcd}$$

MTraS shape modelled by $\sqcup \ltimes \mathbb{I}$ reconciles:

- Hom as a **contra-/covariant bifunctor**,
- Hom as a **constrained function type**.

$\mathbb{I}$ as an MTraS-shape is better behaved on $\bowtie$:

Twisted Cube Category $\bowtie$ [PK20]

(Roughly) the subcategory of NEFinLinOrd (or Δ) generated by $\top$ and $\sqcup \ltimes \mathbb{I}$.

➔ Use $\bowtie$ instead of Δ.

ℹ Pinyo & Kraus carve $\bowtie$ out of **graph** category.

Jet Set of dimension n

Set equipped with n Prop-valued **jet-relations** $\rightarrow\!\triangleright_i$ such that:

- $\rightarrow\!\triangleright_i$ is **reflexive**
- $\rightarrow\!\triangleright_i$ implies $\triangleleft\!\!\rightarrow\!\triangleright_{i+1}$

- **Intervals** $(\!|\rightarrow\!\triangleright_i|\!)$
- **Twisted prism** functor $\sqcup \ltimes (\!|\rightarrow\!\triangleright_i|\!)$ only **op**s degree i
- **Jet cubes** are generated by $\top$ and $\sqcup \ltimes (\!|\rightarrow\!\triangleright_i|\!)$
  - What is a morphism of jet cubes?

Jet Set of dimension n

Set equipped with n Prop-valued **jet-relations** $\multimapdotbothB_i$ such that:

- $\multimapdotbothB_i$ is **reflexive**
- $\multimapdotbothB_i$ implies $\multimapbothvert_{i+1}$

- **Intervals** $(\!|\multimapdotbothB_i|\!)$
- **Twisted prism** functor $\sqcup \ltimes (\!|\multimapdotbothB_i|\!)$ only **op**s degree i
- **Jet cubes** are generated by $\top$ and $\sqcup \ltimes (\!|\multimapdotbothB_i|\!)$
  - What is a morphism of jet cubes?

Jet Set of dimension n

Set equipped with n Prop-valued **jet-relations** $\twoheadrightarrow_i$ such that:

- $\twoheadrightarrow_i$ is **reflexive**
- $\twoheadrightarrow_i$ implies $\leftrightarrow_{i+1}$

- **Intervals** $(\!|\twoheadrightarrow_i|\!)$
- **Twisted prism** functor $\sqcup \ltimes (\!|\twoheadrightarrow_i|\!)$ only **op**s degree i
- **Jet cubes** are generated by $\top$ and $\sqcup \ltimes (\!|\twoheadrightarrow_i|\!)$
  - ? What is a morphism of jet cubes?

$$
\begin{tikzcd}
\mathsf{JetCube}_n \arrow[rr, bend left] \arrow[dr, dotted, "\mathsf{ForgetDeg}"'] & & \mathsf{JetSet}_n \arrow[d, "U"] \\
& \mathsf{Cube}^{⌑}_M \arrow[r, dotted, "\mathsf{Endpoints}"'] \arrow[u, phantom, "\bullet"] & \mathsf{Set}
\end{tikzcd}
$$

- What **interval operations** do you want? ➔ $\mathsf{Cube}^{\boxslash}_M \cong \mathsf{Kleisli}(M)^{\mathsf{op}}$
- Do you want diagonals? ➔ $⌑ \in \{\square, \boxslash\}$
- Turns out only $\mathsf{Cube}^{\square}_{0,1,\neg}$ and $\mathsf{Cube}^{\square}_{\mathsf{FreeBoolAlg}}$ really work.

The Category of **Jet Cubes**

$$\begin{tikzcd}
\mathrm{JetCube}_n \arrow[rr, bend left] \arrow[dr, dotted, "\mathrm{ForgetDeg}"'] & & \mathrm{JetSet}_n \arrow[d, "U"] \\
& \mathrm{Cube}_M^{\text{⌑}} \arrow[r, dotted, "\mathrm{Endpoints}"'] & \mathrm{Set}
\end{tikzcd}$$

- What **interval operations** do you want? ➔ $\mathrm{Cube}_M^{\boxslash} \cong \mathrm{Kleisli}(M)^{\mathrm{op}}$
- Do you want diagonals? ➔ $\text{⌑} \in \{\square, \boxslash\}$
- Turns out only $\mathrm{Cube}_{0,1,\neg}^{\square}$ and $\mathrm{Cube}_{\mathrm{FreeBoolAlg}}^{\square}$ really work.

The Category of **Jet Cubes**

$$
\begin{tikzcd}
\mathrm{JetCube}_n \arrow[rrd, bend left] \arrow[rd, dotted, "\mathrm{ForgetDeg}"'] & & \\
 & & \mathrm{JetSet}_n \arrow[d, "U"] \\
 & \mathrm{Cube}^{\text{⌑}}_M \arrow[r, "\mathrm{Endpoints}"'] & \mathrm{Set}
\end{tikzcd}
$$

- What **interval operations** do you want? ➔ $\mathrm{Cube}^{\boxslash}_M \cong \mathrm{Kleisli}(M)^{\mathrm{op}}$
- Do you want diagonals? ➔ $\text{⌑} \in \{\square, \boxslash\}$
- Turns out only $\mathrm{Cube}^{\square}_{0,1,\neg}$ and $\mathrm{Cube}^{\square}_{\mathrm{FreeBoolAlg}}$ really work.

The Category of **Jet Cubes**

$$\begin{tikzcd} \mathrm{JetCube}_n \arrow[r] \arrow[d, "\mathrm{ForgetDeg}"'] & \mathrm{JetSet}_n \arrow[d, "U"] \\ \mathrm{Cube}^{⌑}_M \arrow[r, "\mathrm{Endpoints}"'] & \mathrm{Set} \end{tikzcd}$$

- What **interval operations** do you want? ➔ $\mathrm{Cube}^{\boxslash}_M \cong \mathrm{Kleisli}(M)^{\mathrm{op}}$
- Do you want diagonals? ➔ $⌑ \in \{\Box, \boxslash\}$
- Turns out only $\mathrm{Cube}^{\Box}_{0,1,\neg}$ and $\mathrm{Cube}^{\Box}_{\mathrm{FreeBoolAlg}}$ really work.

The Category of **Jet Cubes**

$$
\begin{tikzcd}
\mathrm{JetCube}_n \arrow[dr, hook] \arrow[drr, bend left] \arrow[ddr, bend right, "\mathrm{ForgetDeg}"'] & & \\
& \bullet \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathrm{JetSet}_n \arrow[d, "U"] \\
& \mathrm{Cube}^{⌑}_{M} \arrow[r, "\mathrm{Endpoints}"'] & \mathrm{Set}
\end{tikzcd}
$$

- What **interval operations** do you want? ➔ $\mathrm{Cube}^{\boxslash}_{M} \cong \mathrm{Kleisli}(M)^{\mathrm{op}}$
- Do you want diagonals? ➔ $⌑ \in \{\square, \boxslash\}$
- Turns out only $\mathrm{Cube}^{\square}_{0,1,\neg}$ and $\mathrm{Cube}^{\square}_{\mathrm{FreeBoolAlg}}$ really work.

The Category of **Jet Cubes**

$$
\begin{tikzcd}
\mathsf{JetCube}_n \arrow[dr, hook] \arrow[drr, bend left] \arrow[ddr, bend right, "\mathsf{ForgetDeg}"'] & & \\
& \bullet \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathsf{JetSet}_n \arrow[d, "U"] \\
& \mathsf{Cube}^{\text{⌑}}_M \arrow[r, "\mathsf{Endpoints}"'] & \mathsf{Set}
\end{tikzcd}
$$

- What **interval operations** do you want? ➔ $\mathsf{Cube}^{\boxslash}_M \cong \mathsf{Kleisli}(M)^{\mathsf{op}}$
- Do you want diagonals? ➔ $\text{⌑} \in \{\square, \boxslash\}$
- Turns out only $\mathsf{Cube}^{\square}_{0,1,\neg}$ and $\mathsf{Cube}^{\square}_{\mathsf{FreeBoolAlg}}$ really work.

When is a morphism of **cubes** a morphism of **jet cubes**?

TERMINAL

$$\frac{}{\vdash () : V \to ()}$$

SRC:FWD

$$\frac{\vdash \varphi : V \to \mathsf{Op}_i^{\boxdot}(W)}{\vdash (\varphi, 0/\mathbf{i}) : V \to (W, \mathbf{i} : \llparenthesis \multimap_i \rrparenthesis)}$$

SRC:BCK

$$\frac{\vdash \varphi : V \to \mathsf{Op}_i^{\boxdot}(W)}{\vdash (\varphi, 1/\mathbf{i}) : V \to (W, \mathbf{i} : \llparenthesis \multimapinv_i \rrparenthesis)}$$

TGT:FWD

$$\frac{\vdash \varphi : V \to W}{\vdash (\varphi, 1/\mathbf{i}) : V \to (W, \mathbf{i} : \llparenthesis \multimap_i \rrparenthesis)}$$

TGT:BCK

$$\frac{\vdash \varphi : V \to W}{\vdash (\varphi, 0/\mathbf{i}) : V \to (W, \mathbf{i} : \llparenthesis \multimapinv_i \rrparenthesis)}$$

INV:FWD

$$\frac{\vdash (\varphi, t/\mathbf{i}) : V \to (W, \mathbf{i} : \llparenthesis \multimapinv_i \rrparenthesis)}{\vdash (\varphi, \neg t/\mathbf{i}) : V \to (W, \mathbf{i} : \llparenthesis \multimap_i \rrparenthesis)}$$

INV:BCK

$$\frac{\vdash (\varphi, t/\mathbf{i}) : V \to (W, \mathbf{i} : \llparenthesis \multimap_i \rrparenthesis)}{\vdash (\varphi, \neg t/\mathbf{i}) : V \to (W, \mathbf{i} : \llparenthesis \multimapinv_i \rrparenthesis)}$$

PRISM:FWD

$$\frac{\vdash \varphi : V \to W}{\vdash (\varphi, \mathbf{i}/\mathbf{i}) : (V, \mathbf{i} : \llparenthesis \multimap_i \rrparenthesis) \to (W, \mathbf{i} : \llparenthesis \multimap_i \rrparenthesis)}$$

PRISM:BCK

$$\frac{\vdash \varphi : V \to W}{\vdash (\varphi, \mathbf{i}/\mathbf{i}) : (V, \mathbf{i} : \llparenthesis \multimapinv_i \rrparenthesis) \to (W, \mathbf{i} : \llparenthesis \multimapinv_i \rrparenthesis)}$$

SYMMETRIZE

$$\frac{\vdash \varphi : \mathsf{FSym}_i^{\boxdot} V \to W}{\vdash \varphi : V \to \mathsf{USym}_i^{\boxdot} W}$$

WKN

$$\frac{\vdash \varphi : \mathsf{SymCl}_i^{\boxdot}(V) \to W \qquad R \in \{\multimap, \multimapinv, \multimapboth\}}{\vdash (\varphi, \mathbf{i}/\oslash) : (V, \mathbf{i} : \llparenthesis R_i \rrparenthesis) \to W}$$

EXCHANGE

$$\frac{\vdash \varphi : (V, \mathbf{j} : \llparenthesis \multimapboth_i \rrparenthesis, U_1, \mathbf{i} : \llparenthesis \multimapboth_i \rrparenthesis, U_2) \to W}{\vdash \varphi : (V, \mathbf{i} : \llparenthesis \multimapboth_i \rrparenthesis, U_1, \mathbf{j} : \llparenthesis \multimapboth_i \rrparenthesis, U_2) \to W}$$

CONCURSOR

$$\frac{\begin{array}{l} P \in \{\multimap, \multimapinv, \multimapboth\} \qquad Q \in \{\multimap, \multimapinv\} \qquad j > i \\ \vdash \varphi : \mathsf{SymCl}_i^{\boxdot}(\mathsf{SymCl}_j^{\boxdot} U, V) \to W \end{array}}{\vdash (\varphi, \mathbf{j}/\mathbf{i}) : (U, \mathbf{j} : \llparenthesis P_j \rrparenthesis, V) \to (W, \mathbf{i} : \llparenthesis Q_i \rrparenthesis)}$$

CONN:PRISM:SRC-NEUTRAL

$$\frac{\begin{array}{l} (Q, \Diamond) \in \{(\multimap, \vee), (\multimapinv, \wedge)\} \\ \vdash \varphi : \mathsf{SymCl}_i^{\boxdot} V \to W \\ \vdash (\varphi, t/\mathbf{i}) : \mathsf{Op}_i^{\boxdot} V \to (W, \mathbf{i} : \llparenthesis Q_i \rrparenthesis) \end{array}}{\vdash (\varphi, t \Diamond \mathbf{i}/\mathbf{i}) : (V, \mathbf{i} : \llparenthesis Q_i \rrparenthesis) \to (W, \mathbf{i} : \llparenthesis Q_i \rrparenthesis)}\,\mathsf{Boo}$$

CONN:PRISM:TGT-NEUTRAL

$$\frac{\begin{array}{l} (Q, \Diamond) \in \{(\multimap, \wedge), (\multimapinv, \vee)\} \\ \vdash \varphi : \mathsf{SymCl}_i^{\boxdot} V \to W \\ \vdash (\varphi, t/\mathbf{i}) : V \to (W, \mathbf{i} : \llparenthesis Q_i \rrparenthesis) \end{array}}{\vdash (\varphi, t \Diamond \mathbf{i}/\mathbf{i}) : (V, \mathbf{i} : \llparenthesis Q_i \rrparenthesis) \to (W, \mathbf{i} : \llparenthesis Q_i \rrparenthesis)}\,\mathsf{Boo}$$

CONN:PRISM-INV:SRC-NEUTRAL

$$\frac{\begin{array}{l} (Q, \Diamond, P) \in \{(\multimap, \vee, \multimapinv), (\multimapinv, \wedge, \multimap)\} \\ \vdash \varphi : \mathsf{SymCl}_i^{\boxdot} V \to W \\ \vdash (\varphi, t/\mathbf{i}) : \mathsf{Op}_i^{\boxdot} V \to (W, \mathbf{i} : \llparenthesis Q_i \rrparenthesis) \end{array}}{\vdash (\varphi, t \Diamond \neg\mathbf{i}/\mathbf{i}) : (V, \mathbf{i} : \llparenthesis P_i \rrparenthesis) \to (W, \mathbf{i} : \llparenthesis Q_i \rrparenthesis)}\,\mathsf{Boo}$$

CONN:PRISM-INV:TGT-NEUTRAL

$$\frac{\begin{array}{l} (Q, \Diamond, P) \in \{(\multimap, \wedge, \multimapinv), (\multimapinv, \vee, \multimap)\} \\ \vdash \varphi : \mathsf{SymCl}_i^{\boxdot} V \to W \\ \vdash (\varphi, t/\mathbf{i}) : V \to (W, \mathbf{i} : \llparenthesis Q_i \rrparenthesis) \end{array}}{\vdash (\varphi, t \Diamond \neg\mathbf{i}/\mathbf{i}) : (V, \mathbf{i} : \llparenthesis P_i \rrparenthesis) \to (W, \mathbf{i} : \llparenthesis Q_i \rrparenthesis)}\,\mathsf{Boo}$$

CONN:DEGREE-SYMMETRIC

$$\frac{\begin{array}{l} Q \in \{\multimap, \multimapinv\} \qquad \Diamond \in \{\vee, \wedge\} \\ \vdash (\varphi, s/\mathbf{i}) : \mathsf{SymCl}_i^{\boxdot} V \to (W, \mathbf{i} : \llparenthesis Q_i \rrparenthesis) \\ \vdash (\varphi, t/\mathbf{i}) : V \to (W, \mathbf{i} : \llparenthesis Q_i \rrparenthesis) \end{array}}{\vdash (\varphi, t \Diamond s/\mathbf{i}) : V \to (W, \mathbf{i} : \llparenthesis Q_i \rrparenthesis)}\,\mathsf{Boo}$$

Semantic Modalities

⌛ In progress. . .

Three Approaches to the Model

$$\begin{tikzcd}
\text{Tamsamani \& Simpson's model of 2-categories (2-simplicial sets)} \arrow[r, "\text{heterogenize}"'] \arrow[d, "\text{higher dimension}"] & \text{Pro-arrow (pre-)equipments [Woo82,Woo85]} \arrow[d, "\text{higher dimension}"'] & \text{Bridge/path cubical sets (ParamDTT) [NVD17]} \arrow[l, "\text{directify}"] \arrow[d, "\text{higher dimension}"'] \\
\text{Tamsamani \& Simpson's model of } n\text{-categories (} n\text{-simplicial sets) [CL04,Tam99,Sim97]} \arrow[r, "\text{heterogenize}"'] & \textbf{Naturality (Pre)type Theory} & \text{Degrees of Relatedness (RelDTT) [ND18]} \arrow[l, "\text{directify}"]
\end{tikzcd}$$

[CL04] Cheng, E., & Lauda, A. (2004). Higher-Dimensional Categories: an illustrated guide book. http://eugeniacheng.com/guidebook/

[HS97] Hofmann, M., & Streicher, T. (1997). Lifting Grothendieck Universes. Unpublished note.

[NVD17] Nuyts, A., Vezzosi, A., & Devriese, D. (2017). Parametric quantifiers for dependent type theory. PACMPL, 1(ICFP), 32:1-32:29. https://doi.org/10.1145/3110276

[ND18] Nuyts, A., & Devriese, D. (2018). Degrees of Relatedness: A Unified Framework for Parametricity, Irrelevance, Ad Hoc Polymorphism, Intersections, Unions and Algebra in Dependent Type Theory. LICS 2018.

[PK20] Pinyo, G., & Kraus, N. (2020). From Cubes to Twisted Cubes via Graph Morphisms in Type Theory. TYPES 2019 / LIPIcs.

[Sim97] Simpson, C. (1997). Limits in n-categories. ArXiv alg-geom/9708010

[Tam99] Tamsamani, Z. (1999). Sur des notions de n-categorie et n-groupoide non strictes via des ensembles multi-simpliciaux (On the notions of a nonstrict n-category and n-groupoid via multisimplicial sets). K-Theory, 16(1), 51–99.

[Woo82] Wood, R. J. (1982). Abstract pro arrows I. Cahiers de Topologie et Géométrie Différentielle, 23(3), 279–290.

[Woo85] Wood, R. J. (1985). Proarrows II. Cahiers de Topologie et Géométrie Différentielle Catégoriques, 26(2), 135–168.

Thanks!

Questions?

Depth n types

- ***i*-edge** relations $\frown_i$
- **Dependency:**
  $r : a \frown_i^R b$ presumes $R : A \frown_{i+1}^{\mathrm{U}} B$
- **Degradation:**
  $a \frown_i b \Rightarrow a \frown_{i+1} b$
- Modalities change indices:

$$\begin{tikzcd}
a \frown_0 b \arrow[d] & fa \frown_0 fb \arrow[d] \\
a \frown_1 b \arrow[d] \arrow[ur] & fa \frown_1 fb \\
a \frown_2 b \arrow[ur] &
\end{tikzcd}$$

$\boxed{\mathbf{lim}^{\oplus}}$

$$\begin{tikzcd}
a \frown_0 b \arrow[r] \arrow[d] & fa \frown_0 fb \arrow[d] \\
a \frown_1 b \arrow[r] \arrow[d] & fa \frown_1 fb \arrow[d] \\
a \frown_2 b \arrow[r] & fa \frown_2 fb
\end{tikzcd}$$

$\boxed{\mathbf{ftr}^{\oplus}}$

n-equipments

- ***i*-jet** (pro^{i-1}-arrow) relations $\mathrel{-\!\triangleright}_i$
- **Dependency:**
  $j : a \mathrel{-\!\triangleright}_i^J b$ presumes $J : A \mathrel{-\!\triangleright}_{i+1}^{\mathrm{U}} B$
- **Companion / conjoint:**
  $(\ddagger, \dagger) : a \mathrel{-\!\triangleright}_i b \Rightarrow a \mathrel{\triangleleft\!-\!\triangleright}_{i+1} b$
- Modalities change indices & orientation:

$$\begin{tikzcd}
a \mathrel{-\!\triangleright}_0 b \arrow[d, "\circledast"'] & fa \mathrel{-\!\triangleright}_0 fb \arrow[d, "\circledast"] \\
a \mathrel{-\!\triangleright}_1 b \arrow[d, "\circledast"'] \arrow[ur, "\oplus"] & fa \mathrel{-\!\triangleright}_1 fb \\
a \mathrel{-\!\triangleright}_2 b \arrow[ur, "\oplus"] &
\end{tikzcd}$$

$\boxed{\mathbf{lim}^{\ominus}}$

$$\begin{tikzcd}
a \mathrel{-\!\triangleright}_0 b \arrow[r, "\oplus"] \arrow[d, "\circledast"'] & fa \mathrel{-\!\triangleright}_0 fb \arrow[d, "\circledast"] \\
a \mathrel{-\!\triangleright}_1 b \arrow[r, "\ominus"] \arrow[d, "\circledast"'] & fa \mathrel{-\!\triangleright}_1 fb \arrow[d, "\circledast"] \\
a \mathrel{-\!\triangleright}_2 b \arrow[r, "\oplus"] & fa \mathrel{-\!\triangleright}_2 fb
\end{tikzcd}$$

$\boxed{\mathbf{ftr}^{\ominus}}$

Depth n types

- ***i*-edge** relations $\frown_i$
- **Dependency:**
  $r : a \frown_i^{R} b$ presumes $R : A \frown_{i+1}^{\mathrm{U}} B$
- **Degradation:**
  $a \frown_i b \Rightarrow a \frown_{i+1} b$
- Modalities change indices:

$$\begin{tikzcd}
a \frown_0 b \arrow[d] & fa \frown_0 fb \arrow[d] \\
a \frown_1 b \arrow[d] \arrow[ur] & fa \frown_1 fb \\
a \frown_2 b \arrow[ur] &
\end{tikzcd}$$

$\boxed{\mathbf{lim}^{\oplus}}$

$$\begin{tikzcd}
a \frown_0 b \arrow[r] \arrow[d] & fa \frown_0 fb \arrow[d] \\
a \frown_1 b \arrow[r] \arrow[d] & fa \frown_1 fb \arrow[d] \\
a \frown_2 b \arrow[r] & fa \frown_2 fb
\end{tikzcd}$$

$\boxed{\mathbf{ftr}^{\oplus}}$

n-equipments

- ***i*-jet** (pro^{i-1}-arrow) relations $\rightarrowtriangle_i$
- **Dependency:**
  $j : a \rightarrowtriangle_i^{J} b$ presumes $J : A \rightarrowtriangle_{i+1}^{\mathrm{U}} B$
- **Companion / conjoint:**
  $(\ddagger, \dagger) : a \rightarrowtriangle_i b \Rightarrow a \leftrightarrowtriangle_{i+1} b$
- Modalities change indices & orientation:

$$\begin{tikzcd}
a \rightarrowtriangle_0 b \arrow[d, "\circledast"'] & fa \rightarrowtriangle_0 fb \arrow[d, "\circledast"] \\
a \rightarrowtriangle_1 b \arrow[d, "\circledast"'] \arrow[ur, "\oplus"] & fa \rightarrowtriangle_1 fb \\
a \rightarrowtriangle_2 b \arrow[ur, "\oplus"] &
\end{tikzcd}$$

$\boxed{\mathbf{lim}^{\ominus}}$

$$\begin{tikzcd}
a \rightarrowtriangle_0 b \arrow[r, "\oplus"] \arrow[d, "\circledast"'] & fa \rightarrowtriangle_0 fb \arrow[d, "\circledast"] \\
a \rightarrowtriangle_1 b \arrow[r, "\ominus"] \arrow[d, "\circledast"'] & fa \rightarrowtriangle_1 fb \arrow[d, "\circledast"] \\
a \rightarrowtriangle_2 b \arrow[r, "\oplus"] & fa \rightarrowtriangle_2 fb
\end{tikzcd}$$

$\boxed{\mathbf{ftr}^{\ominus}}$

Depth n types

- ***i*-edge** relations $\frown_i$
- **Dependency:**
  $r : a \frown_i^R b$ presumes $R : A \frown_{i+1}^{\mathrm{U}} B$
- **Degradation:**
  $a \frown_i b \Rightarrow a \frown_{i+1} b$
- Modalities change indices:

$$\begin{tikzcd}
a \frown_0 b \arrow[d] & fa \frown_0 fb \arrow[d] \\
a \frown_1 b \arrow[d] \arrow[ur] & fa \frown_1 fb \\
a \frown_2 b \arrow[ur] &
\end{tikzcd}
\qquad
\begin{tikzcd}
a \frown_0 b \arrow[r] \arrow[d] & fa \frown_0 fb \arrow[d] \\
a \frown_1 b \arrow[r] \arrow[d] & fa \frown_1 fb \arrow[d] \\
a \frown_2 b \arrow[r] & fa \frown_2 fb
\end{tikzcd}$$

$\boxed{\mathbf{lim}^{\oplus}}$ $\boxed{\mathbf{ftr}^{\oplus}}$

n-equipments

- ***i*-jet** (pro^{i-1}-arrow) relations $\mathrel{-\!\triangleright}_i$
- **Dependency:**
  $j : a \mathrel{-\!\triangleright}_i^J b$ presumes $J : A \mathrel{-\!\triangleright}_{i+1}^{\mathrm{U}} B$
- **Companion / conjoint:**
  $(\ddagger, \dagger) : a \mathrel{-\!\triangleright}_i b \Rightarrow a \mathrel{\triangleleft\!-\!\triangleright}_{i+1} b$
- Modalities change indices & orientation:

$$\begin{tikzcd}
a \mathrel{-\!\triangleright}_0 b \arrow[d, "\circledast"'] & fa \mathrel{-\!\triangleright}_0 fb \arrow[d, "\circledast"] \\
a \mathrel{-\!\triangleright}_1 b \arrow[d, "\circledast"'] \arrow[ur, "\oplus"] & fa \mathrel{-\!\triangleright}_1 fb \\
a \mathrel{-\!\triangleright}_2 b \arrow[ur, "\oplus"] &
\end{tikzcd}
\qquad
\begin{tikzcd}
a \mathrel{-\!\triangleright}_0 b \arrow[r, "\oplus"] \arrow[d, "\circledast"'] & fa \mathrel{-\!\triangleright}_0 fb \arrow[d, "\circledast"] \\
a \mathrel{-\!\triangleright}_1 b \arrow[r, "\ominus"] \arrow[d, "\circledast"'] & fa \mathrel{-\!\triangleright}_1 fb \arrow[d, "\circledast"] \\
a \mathrel{-\!\triangleright}_2 b \arrow[r, "\oplus"] & fa \mathrel{-\!\triangleright}_2 fb
\end{tikzcd}$$

$\boxed{\mathbf{lim}^{\ominus}}$ $\boxed{\mathbf{ftr}^{\ominus}}$

Depth n types

- ***i*-edge** relations $\frown_i$
- **Dependency:**
  $r : a \frown_i^{R} b$ presumes $R : A \frown_{i+1}^{\mathrm{U}} B$
- **Degradation:**
  $a \frown_i b \Rightarrow a \frown_{i+1} b$
- Modalities change indices:

$$\begin{tikzcd}
a \frown_0 b \arrow[d] & fa \frown_0 fb \arrow[d] \\
a \frown_1 b \arrow[d] \arrow[ur] & fa \frown_1 fb \\
a \frown_2 b \arrow[ur] & \boxed{\mathbf{lim}^{\oplus}}
\end{tikzcd}
\qquad
\begin{tikzcd}
a \frown_0 b \arrow[r] \arrow[d] & fa \frown_0 fb \arrow[d] \\
a \frown_1 b \arrow[r] \arrow[d] & fa \frown_1 fb \arrow[d] \\
a \frown_2 b \arrow[r] & fa \frown_2 fb
\end{tikzcd}
\quad \boxed{\mathbf{ftr}^{\oplus}}$$

n-equipments

- ***i*-jet** (pro^{i-1}-arrow) relations $\rightarrowtriangle_i$
- **Dependency:**
  $j : a \rightarrowtriangle_i^{J} b$ presumes $J : A \rightarrowtriangle_{i+1}^{\mathrm{U}} B$
- **Companion / conjoint:**
  $(\ddagger, \dagger) : a \rightarrowtriangle_i b \Rightarrow a \leftrightarrowtriangle_{i+1} b$
- Modalities change indices & orientation:

$$\begin{tikzcd}
a \rightarrowtriangle_0 b \arrow[d, "\circledast"'] & fa \rightarrowtriangle_0 fb \arrow[d, "\circledast"] \\
a \rightarrowtriangle_1 b \arrow[d, "\circledast"'] \arrow[ur, "\oplus"] & fa \rightarrowtriangle_1 fb \\
a \rightarrowtriangle_2 b \arrow[ur, "\oplus"] & \boxed{\mathbf{lim}^{\ominus}}
\end{tikzcd}
\qquad
\begin{tikzcd}
a \rightarrowtriangle_0 b \arrow[r, "\oplus"] \arrow[d, "\circledast"'] & fa \rightarrowtriangle_0 fb \arrow[d, "\circledast"] \\
a \rightarrowtriangle_1 b \arrow[r, "\ominus"] \arrow[d, "\circledast"'] & fa \rightarrowtriangle_1 fb \arrow[d, "\circledast"] \\
a \rightarrowtriangle_2 b \arrow[r, "\oplus"] & fa \rightarrowtriangle_2 fb
\end{tikzcd}
\quad \boxed{\mathbf{ftr}^{\ominus}}$$