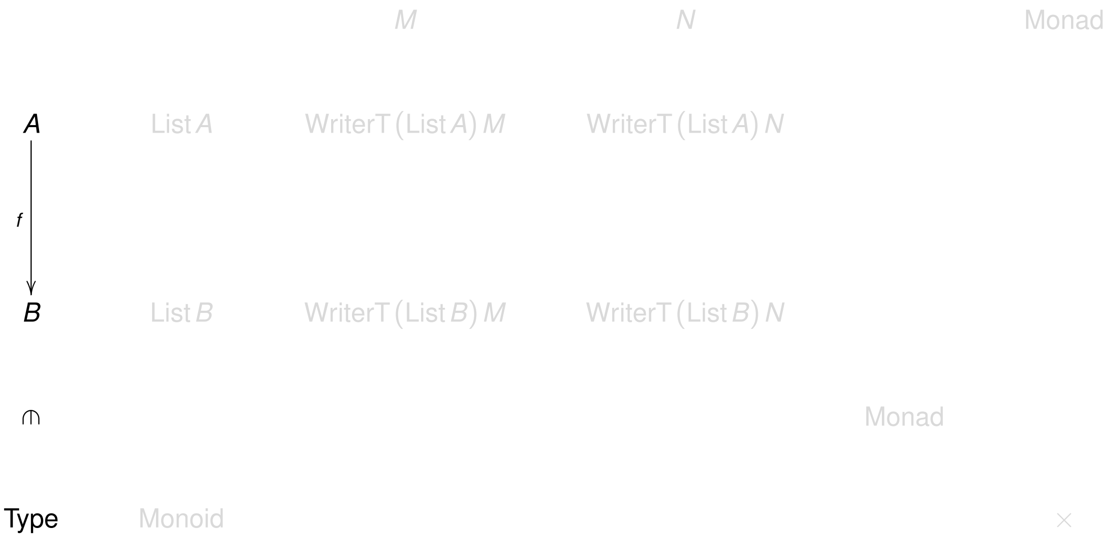
Higher Pro-arrows: Towards a Model for Naturality Pretype Theory

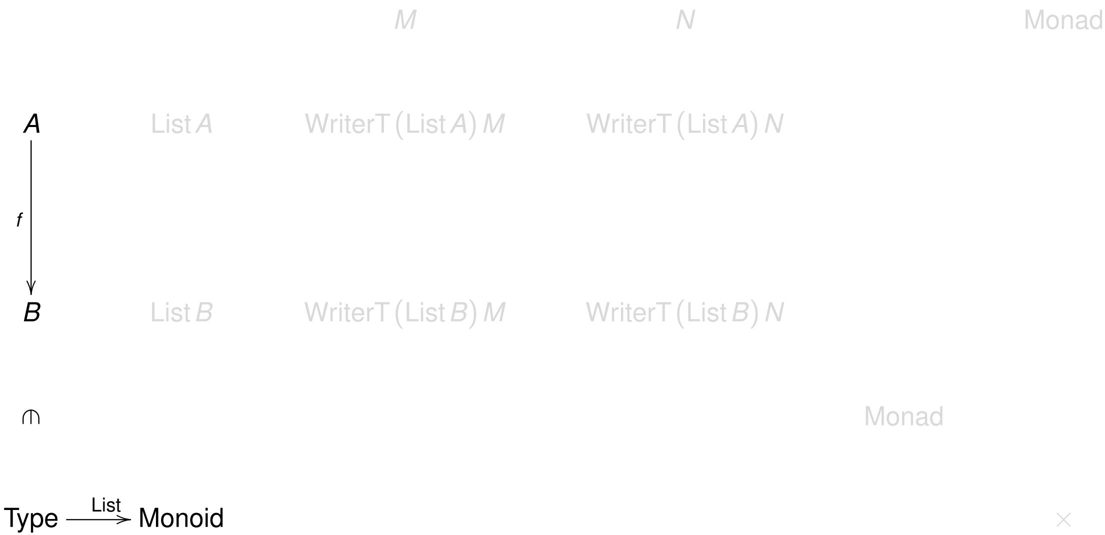
Andreas Nuyts

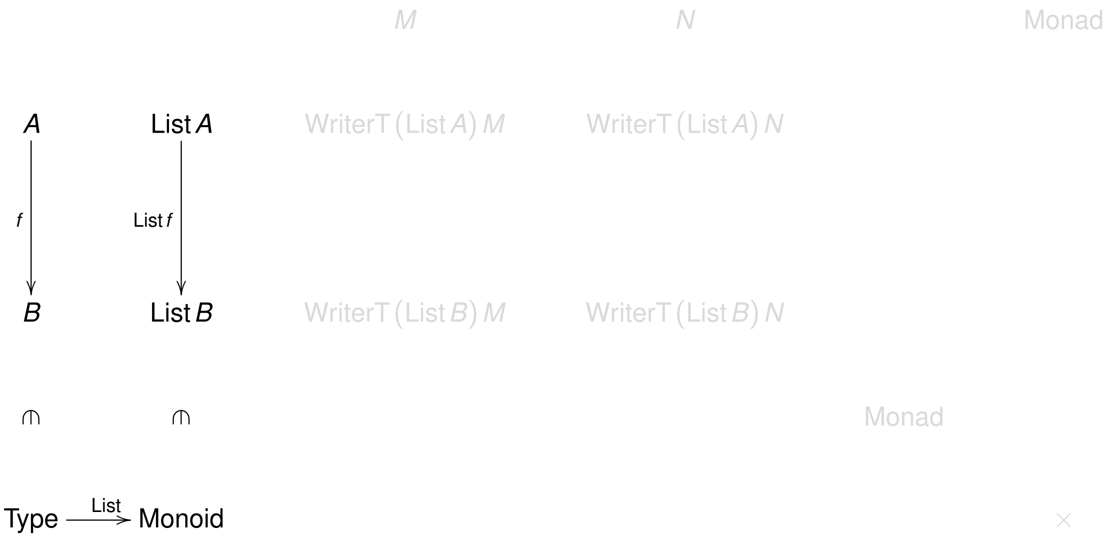
KU Leuven, Belgium

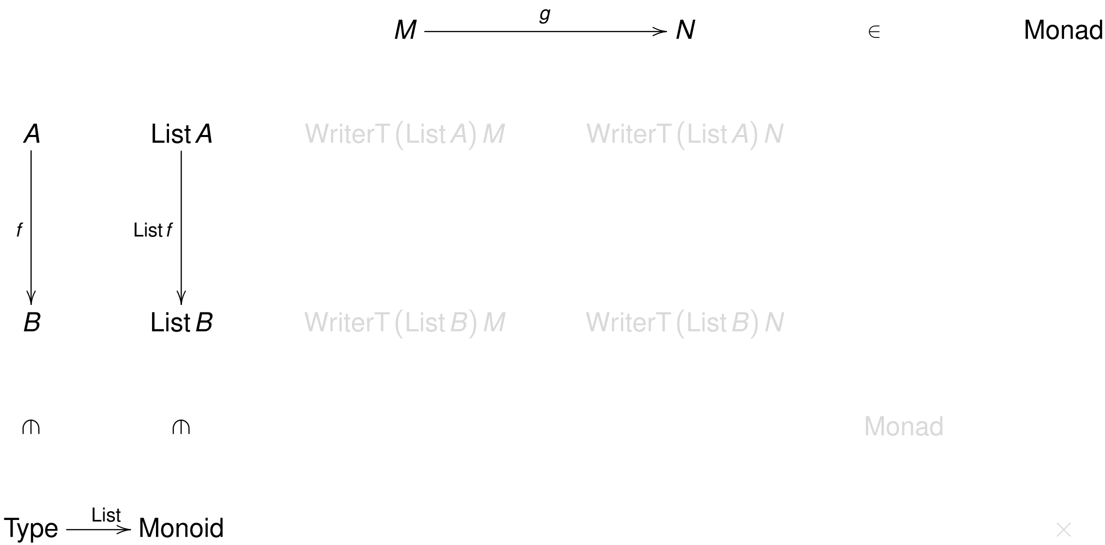
HoTT/UF ’23
Vienna, Austria
April 23, 2023

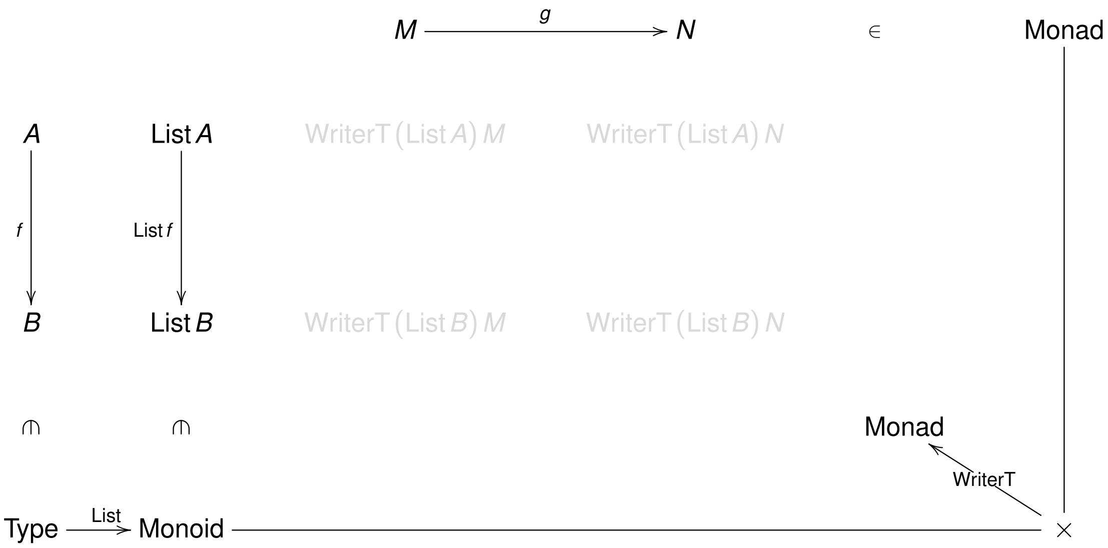
Naturality TT: **Why?** (And what?)
Example problem in verified functional programming

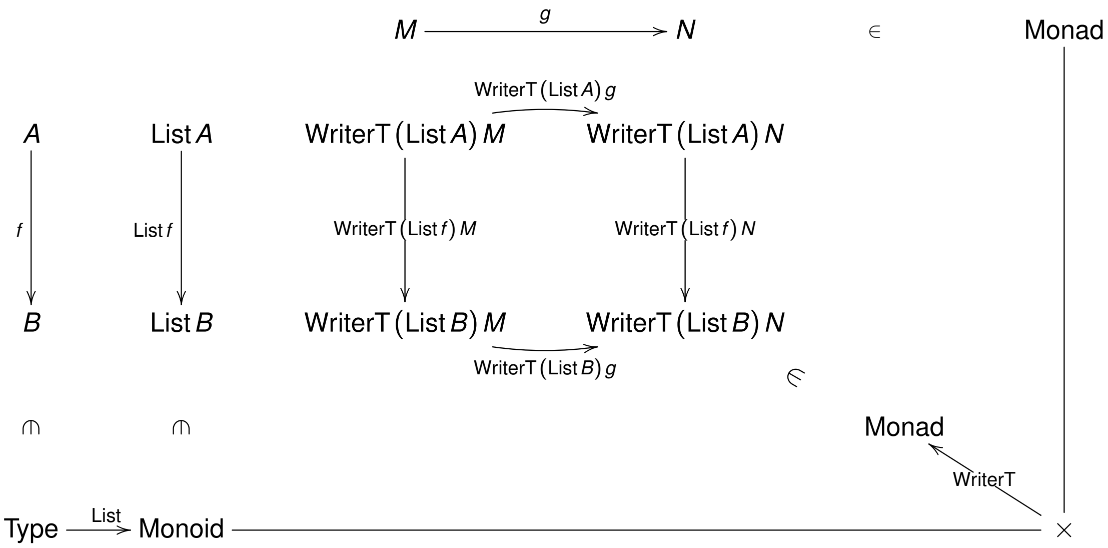

M
N
Monad
A
List A
WriterT (List A) M
WriterT (List A) N
f
B
List B
WriterT (List B) M
WriterT (List B) N
⋔
Monad
Type
Monoid

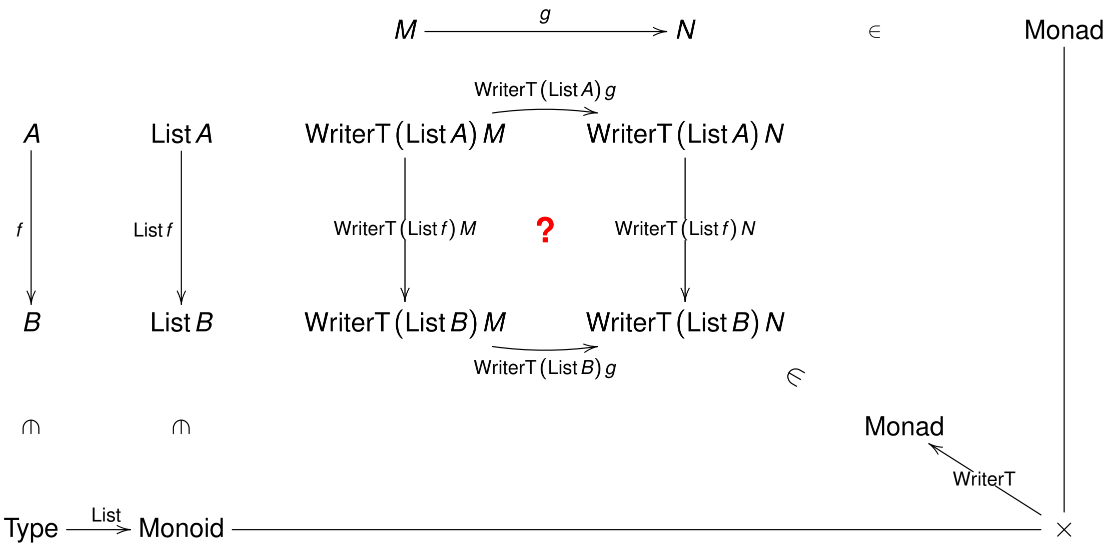

M
N
Monad
A
List A
WriterT (List A) M
WriterT (List A) N
f
B
List B
WriterT (List B) M
WriterT (List B) N
⋔
Monad
Type
List
Monoid
×


M
N
Monad
A
List A
WriterT (List A) M
WriterT (List A) N
f
List f
B
List B
WriterT (List B) M
WriterT (List B) N
Monad
Type
List
Monoid


M
g
N
∈
Monad
A
List A
WriterT (List A) M
WriterT (List A) N
f
List f
B
List B
WriterT (List B) M
WriterT (List B) N
Monad
Type
List
Monoid


M
g
N
∈
Monad
A
List A
WriterT (List A) M
WriterT (List A) N
f
List f
B
List B
WriterT (List B) M
WriterT (List B) N
⋔
⋔
Monad
WriterT
Type
List
Monoid
×


M
g
N
∈
Monad
WriterT (List A) g
A
List A
WriterT (List A) M
WriterT (List A) N
f
List f
WriterT (List f) M
WriterT (List f) N
B
List B
WriterT (List B) M
WriterT (List B) N
WriterT (List B) g
∈
∈
∈
Monad
WriterT
Type
List
Monoid
×


M
g
N
∈
Monad
WriterT (List A) g
A
List A
WriterT (List A) M
WriterT (List A) N
f
List f
WriterT (List f) M
?
WriterT (List f) N
B
List B
WriterT (List B) M
WriterT (List B) N
WriterT (List B) g
∈
⋔
⋔
Monad
WriterT
Type
List
Monoid
×

In plain DTT

Functoriality of List : Type $\to$ Monoid:

- Object action: $(\text{List}\, A, [], ++)$
- 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 induction)**

  + 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

In plain DTT

Functoriality of List : Type $\to$ Monoid:

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

  + 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

In plain DTT

Functoriality of List : Type $\to$ Monoid:

- Object action: $(\text{List}\, A, [], ++)$
- 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 induction)**

  + 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

In parametric DTT

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 induction)**

  + functor laws **(by induction)**

Functoriality of
WriterT : Monoid $\to$ MonadTrans

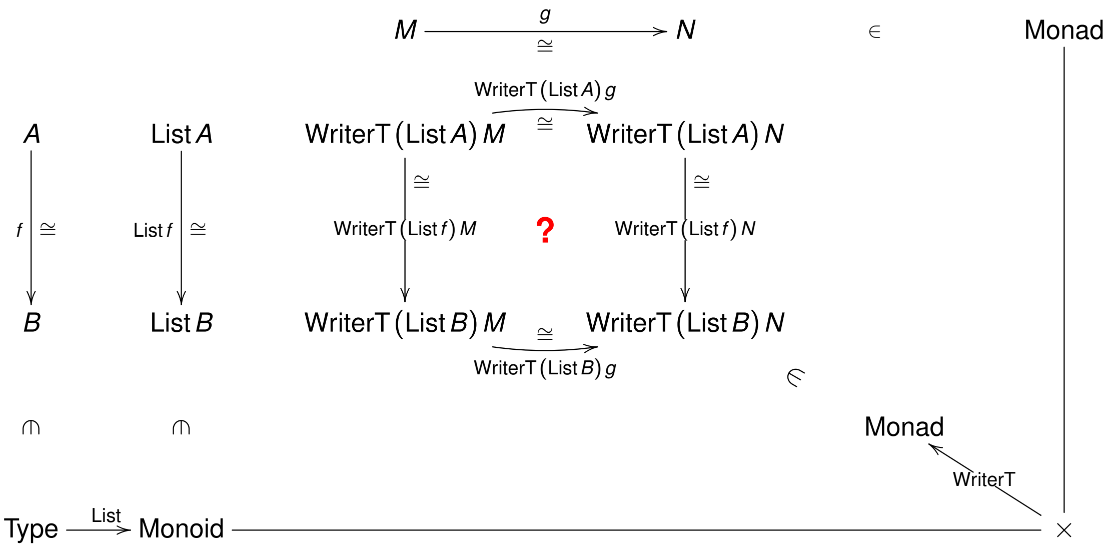
- Object action: WriterT $W \in$ MonadTrans
  - Object action: WriterT $W\,M \in$ Monad
    - Object action: Define WriterT $W\,M\,A$
    - Functorial action WriterT $W\,M\,f$
      + functor laws
    - return & bind + 🎁 naturality

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

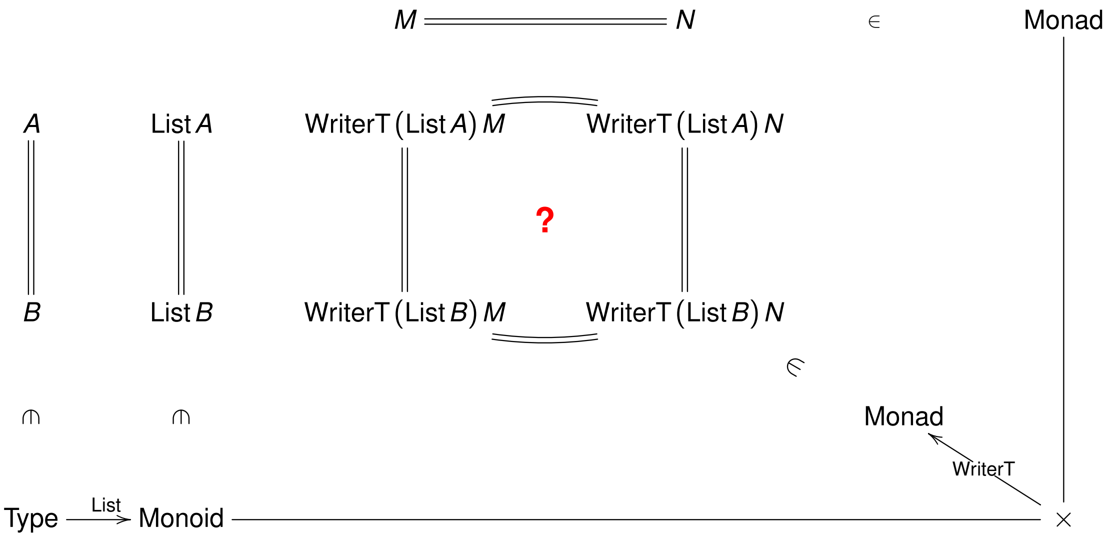
  - Functorial action WriterT $W\,g$
    - Respects return & bind

    + 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



M
g
≅
N
∈
Monad
WriterT (List A) g
A
List A
WriterT (List A) M
≅
WriterT (List A) N
f
≅
List f
≅
≅
WriterT (List f) M
?
≅
WriterT (List f) N
B
List B
WriterT (List B) M
≅
WriterT (List B) N
WriterT (List B) g
∈
∈
∈
Monad
WriterT
Type
List
Monoid
×


M
N
∈
Monad
A
List A
WriterT (List A) M
WriterT (List A) N
?
B
List B
WriterT (List B) M
WriterT (List B) N
∈
⋔
⋔
Monad
WriterT
Type
List
Monoid
×

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

Functoriality of List : Type $\to$ Monoid:

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

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

Functoriality of

WriterT : Monoid $\to$ MonadTrans

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

      \+ functor laws
    - return & bind + naturality

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

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

    \+ functor laws
  - lift : $M \to \text{WriterT}\, W\, M$ + naturality
    - Respects return & bind
- Functorial action:
  $\text{WriterT}\, h : \text{WriterT}\, V \cong \text{WriterT}\, W$
  - $\text{WriterT}\, h\, M\, A$
    - 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: (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 $W\,M\,A$: Monad is **covariant** w.r.t.

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

ReaderT $R\,M\,A$ 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

WriterT $W\,M\,A$: Monad is **covariant** w.r.t.

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

ReaderT $R\,M\,A$ 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

WriterT $W\,M\,A$: Monad is **covariant** w.r.t.

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

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

- R : Type

return : $A \rightarrow$ 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 $W\, M\, A$: Monad is **covariant** w.r.t.

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

ReaderT $R\, M\, A$ 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

WriterT $W\,M\,A$: Monad is **covariant** w.r.t.

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

ReaderT $R\,M\,A$ 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

WriterT $W\,M\,A$: Monad is **covariant** w.r.t.

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

ReaderT $R\,M\,A$ 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

WriterT $W\,M\,A$: Monad is **covariant** w.r.t.

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

ReaderT $R\,M\,A$ 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

Pretypes: A Note on Fibrancy

A presheaf model of DTT can account for the **existence** of **paths/morphisms/bridges/...**

Fibrant types have **operations** for these:

Directed	
functorial	Transport along morphisms
Segal	Composition of morphisms
Rezk	Isomorphism-path univalence

HoTT	
Kan	Comp. of & transp. along paths

Param'ty	
discrete	Homog. bridges express equality
...	...

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 [BT21, Nuy20]
  - Amazing right adjoint [LOPS18] & Transpension [ND21]
  - Internal fibrant replacement monad [Nuy20]

$\Rightarrow$ It's a **pretype** system

A note on fibrancy

A presheaf model of DTT can account for the **existence** of **paths/morphisms/bridges/...**

Fibrant types have **operations** for these:

We **ignore** fibrancy for now:

- Functoriality & Segal fibrancy are brittle
  ⇒ need to consider pretypes anyway
- There are promising techniques for defining fibrancy internally:
  - Contextual fibrancy [BT21, Nuy20]
  - Amazing right adjoint [LOPS18] & Transpension [ND21]
  - Internal fibrant replacement monad [Nuy20]

⇒ It's a **pretype** system

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

A note on fibrancy

A presheaf model of DTT can account for the **existence** of **paths/morphisms/bridges/...**

Fibrant types have **operations** for these:

We **ignore** fibrancy for now:

- Functoriality & Segal fibrancy are brittle
  ⇒ need to consider pretypes anyway
- There are promising techniques for defining fibrancy internally:
  - Contextual fibrancy [BT21, Nuy20]
  - Amazing right adjoint [LOPS18] & Transpension [ND21]
  - Internal fibrant replacement monad [Nuy20]

⇒ It's a **pretype** system

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

A note on fibrancy

A presheaf model of DTT can account for the **existence** of **paths/morphisms/bridges/...**

Fibrant types have **operations** for these:

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

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 [BT21, Nuy20]
  - Amazing right adjoint [LOPS18] & Transpension [ND21]
  - Internal fibrant replacement monad [Nuy20]

$\Rightarrow$ It's a **pretype** system

A note on fibrancy

A presheaf model of DTT can account for the **existence** of **paths/morphisms/bridges/. . .**

Fibrant types have **operations** for these:

We **ignore** fibrancy for now:

- Functoriality & Segal fibrancy are brittle
  ⇒ need to consider pretypes anyway
- There are promising techniques for defining fibrancy internally:
  - Contextual fibrancy [BT21, Nuy20]
  - Amazing right adjoint [LOPS18] & Transpension [ND21]
  - Internal fibrant replacement monad [Nuy20]

⇒ It's a **pretype** system

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

A note on fibrancy

A presheaf model of DTT can account for the **existence** of **paths/morphisms/bridges/. . .**

Fibrant types have **operations** for these:

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 [BT21, Nuy20]
  - Amazing right adjoint [LOPS18] & Transpension [ND21]
  - Internal fibrant replacement monad [Nuy20]

$\Rightarrow$ It's a **pretype** system

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

A note on fibrancy

A presheaf model of DTT can account for the **existence** of **paths/morphisms/bridges/...**

Fibrant types have **operations** for these:

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 [BT21, Nuy20]
  - Amazing right adjoint [LOPS18] & Transpension [ND21]
  - Internal fibrant replacement monad [Nuy20]

$\Rightarrow$ It's a **pretype** system

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

Model-first Approach

The type system emerges from the model:

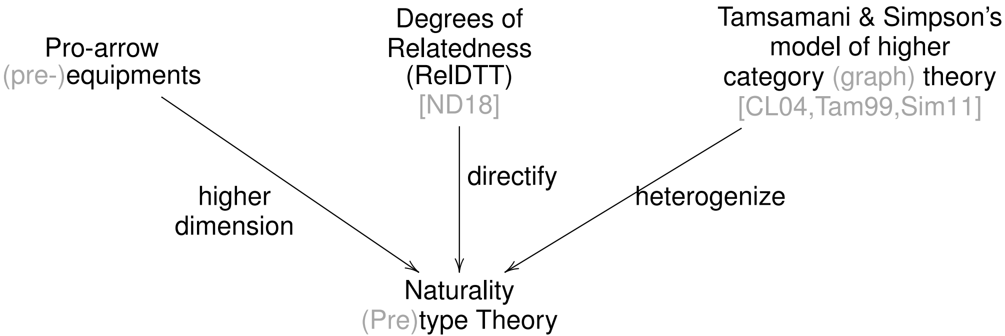
- A diagram of CwFs and adjunctions models an instance of MTT [GKNB20].
- An endofunctor on $\mathscr{W}$ models a substructural shape (e.g. $\mathbb{I}$) in $\mathrm{Psh}(\mathscr{W})$ giving rise to modalities $\exists(i : \mathbb{I}) \dashv ⅎ[i : \mathbb{I}] \dashv \forall(i : \mathbb{I}) \dashv \between[i : \mathbb{I}]$. This is the basis of the modal transpension type system (MTraS) [ND21].

Model-first Approach

The type system emerges from the model:

- A diagram of CwFs and adjunctions models an instance of MTT [GKNB20].
- An endofunctor on $\mathscr{W}$ models a substructural shape (e.g. $\mathbb{I}$) in $\mathrm{Psh}(\mathscr{W})$ giving rise to modalities $\exists(i : \mathbb{I}) \dashv \Finv[i : \mathbb{I}] \dashv \forall(i : \mathbb{I}) \dashv \between[i : \mathbb{I}]$. This is the basis of the modal transpension type system (MTraS) [ND21].

The Model


Pro-arrow
(pre-)equipments
Degrees of
Relatedness
(RelDTT)
[ND18]
Tamsamani & Simpson's
model of higher
category (graph) theory
[CL04,Tam99,Sim11]
higher
dimension
directify
heterogenize
Naturality
(Pre)type Theory

n-Fold Categories

Category

A **category** $\mathscr{C}$ can be defined as a **simplicial set** $\mathscr{C} \in \mathrm{Psh}(\Delta)$ satisfying the **Segal condition**.

n-Fold category

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

Double category

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

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

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

Pretypes!

n-Fold Categories

Category

A **category** $\mathscr{C}$ can be defined as a **simplicial set** $\mathscr{C} \in \mathrm{Psh}(\Delta)$ satisfying the **Segal condition**.

n-Fold category

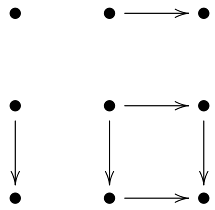
An n**-fold category** $\mathscr{C}$ is an n**-fold simplicial set** $\mathscr{C} \in \mathrm{Psh}(\Delta^n)$ satisfying the **Segal condition** in each dimension.

Double category

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

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

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



Pretypes!

n-Fold Categories

Category

A **category** $\mathscr{C}$ can be defined as a **simplicial set** $\mathscr{C} \in \mathrm{Psh}(\Delta)$ satisfying the **Segal condition**.

n-Fold category

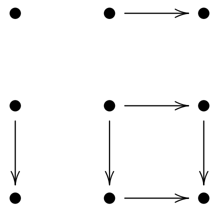
An ***n*-fold category** $\mathscr{C}$ is an ***n*-fold simplicial set** $\mathscr{C} \in \mathrm{Psh}(\Delta^n)$ satisfying the **Segal condition** in each dimension.

Double category

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

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

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



Pretypes!

n-Fold Categories

Category

A **category** $\mathscr{C}$ can be defined as a **simplicial set** $\mathscr{C} \in \mathrm{Psh}(\Delta)$ satisfying the **Segal condition**.

Double category

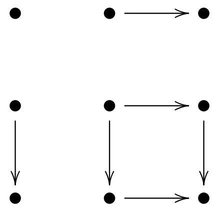
A **double category** $\mathscr{C}$ has:

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

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

n-Fold category

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



Pretypes!

n-Fold Categories

Category

A **category** $\mathscr{C}$ can be defined as a **simplicial set** $\mathscr{C} \in \mathrm{Psh}(\Delta)$ ~~satisfying the~~ ~~**Segal condition**~~.

n-Fold category

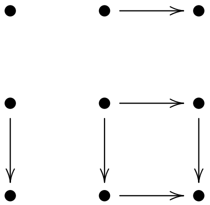
An *n***-fold category** $\mathscr{C}$ is an *n***-fold simplicial set** $\mathscr{C} \in \mathrm{Psh}(\Delta^n)$ ~~satisfying the~~ ~~**Segal condition** in each dimension~~.

Double category

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

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

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



Pretypes!

The Twisted Prism Functor

Δ is a skeleton of FinLinOrd,
hence $\Delta \simeq$ FinLinOrd.

Twisted Prism Functor [PK19]

$\sqcup \ltimes \mathbb{I}$: FinLinOrd $\to$ FinLinOrd :
$W \mapsto W^{\mathrm{op}} \uplus_{<} W$

$$a \longrightarrow b \quad \mapsto \quad \begin{array}{ccc} \iota_0\, a & \longleftarrow & \iota_0\, b \\ \downarrow & & \downarrow \\ \iota_1\, a & \longrightarrow & \iota_1\, b \end{array}$$

An MTraS-shape $\mathbb{I}$ modelled by $\sqcup \ltimes \mathbb{I}$, reconciles the view of Hom as a **contra-/covariant bifunctor** with a view as a directed **path type**.

$\mathbb{I}$ as an MTraS-shape is better behaved on $\bowtie$:

Twisted Cube Category $\bowtie$ [PK19]

(Roughly) the subcategory of FinLinOrd (or Δ) generated by $\top$ and $\sqcup \ltimes \mathbb{I}$.

***n*-Fold category**

An *n*-**fold category** $\mathscr{C}$ is an *n*-**fold twisted cubical set** $\mathscr{C} \in \mathrm{Psh}(\bowtie^n)$ ~~satisfying the **Segal condition** in each dimension.~~

The Twisted Prism Functor

Δ is a skeleton of FinLinOrd,
hence $\Delta \simeq$ FinLinOrd.

Twisted Prism Functor [PK19]

$\sqcup \ltimes \mathbb{I}$: FinLinOrd $\to$ FinLinOrd :
$W \mapsto W^{\mathsf{op}} \uplus_{<} W$

$$a \longrightarrow b \qquad \mapsto \qquad \begin{array}{ccc} \iota_0\, a & \longleftarrow & \iota_0\, b \\ \downarrow & & \downarrow \\ \iota_1\, a & \longrightarrow & \iota_1\, b \end{array}$$

An MTraS-shape $\mathbb{I}$ modelled by $\sqcup \ltimes \mathbb{I}$,
reconciles the view of Hom as a
contra-/covariant bifunctor with a view as a
directed **path type**.

$\mathbb{I}$ as an MTraS-shape is better behaved on $\bowtie$:

Twisted Cube Category $\bowtie$ [PK19]

(Roughly) the subcategory of FinLinOrd (or
Δ) generated by $\top$ and $\sqcup \ltimes \mathbb{I}$.

***n*-Fold category**

An ***n*-fold category** $\mathscr{C}$ is an ***n*-fold twisted cubical set** $\mathscr{C} \in \mathrm{Psh}(\bowtie^n)$ ~~satisfying the~~
~~**Segal condition** in each dimension.~~

The Twisted Prism Functor

Δ is a skeleton of FinLinOrd,
hence $\Delta \simeq$ FinLinOrd.

Twisted Prism Functor [PK19]

$\sqcup \ltimes \mathbb{I}$: FinLinOrd $\to$ FinLinOrd :
$W \mapsto W^{\mathrm{op}} \uplus_{<} W$

$$a \longrightarrow b \quad \mapsto \quad \begin{array}{ccc} \iota_0\, a & \longleftarrow & \iota_0\, b \\ \downarrow & & \downarrow \\ \iota_1\, a & \longrightarrow & \iota_1\, b \end{array}$$

An MTraS-shape $\mathbb{I}$ modelled by $\sqcup \ltimes \mathbb{I}$, reconciles the view of Hom as a **contra-/covariant bifunctor** with a view as a directed **path type**.

$\mathbb{I}$ as an MTraS-shape is better behaved on $\bowtie$:

Twisted Cube Category $\bowtie$ [PK19]

(Roughly) the subcategory of FinLinOrd (or Δ) generated by $\top$ and $\sqcup \ltimes \mathbb{I}$.

***n*-Fold category**

An *n*-**fold category** $\mathscr{C}$ is an *n*-**fold twisted cubical set** $\mathscr{C} \in \mathrm{Psh}(\bowtie^n)$ ~~satisfying the **Segal condition** in each dimension.~~

The Twisted Prism Functor

Δ is a skeleton of FinLinOrd, hence $\Delta \simeq$ FinLinOrd.

Twisted Prism Functor [PK19]

$\textvisiblespace \ltimes \mathbb{I}$: FinLinOrd $\to$ FinLinOrd : $W \mapsto W^{\mathrm{op}} \uplus_< W$

$$a \longrightarrow b \quad \mapsto \quad \begin{array}{ccc} \iota_0\, a & \longleftarrow & \iota_0\, b \\ \downarrow & & \downarrow \\ \iota_1\, a & \longrightarrow & \iota_1\, b \end{array}$$

An MTraS-shape $\mathbb{I}$ modelled by $\textvisiblespace \ltimes \mathbb{I}$, reconciles the view of Hom as a **contra-/covariant bifunctor** with a view as a directed **path type**.

$\mathbb{I}$ as an MTraS-shape is better behaved on $\bowtie$:

Twisted Cube Category $\bowtie$ [PK19]

(Roughly) the subcategory of FinLinOrd (or Δ) generated by $\top$ and $\textvisiblespace \ltimes \mathbb{I}$.

***n*-Fold category**

An *n*-**fold category** $\mathscr{C}$ is an *n*-**fold twisted cubical set** $\mathscr{C} \in \mathrm{Psh}(\bowtie^n)$ ~~satisfying the **Segal condition** in each dimension.~~

The Twisted Prism Functor

Δ is a skeleton of FinLinOrd,
hence $\Delta \simeq$ FinLinOrd.

Twisted Prism Functor [PK19]

$\sqcup \ltimes \mathbb{I}$: FinLinOrd $\to$ FinLinOrd :
$W \mapsto W^{\mathrm{op}} \uplus_{<} W$

$$a \longrightarrow b \quad \mapsto \quad \begin{array}{ccc} \iota_0\, a & \longleftarrow & \iota_0\, b \\ \downarrow & & \downarrow \\ \iota_1\, a & \longrightarrow & \iota_1\, b \end{array}$$

An MTraS-shape $\mathbb{I}$ modelled by $\sqcup \ltimes \mathbb{I}$, reconciles the view of Hom as a **contra-/covariant bifunctor** with a view as a directed **path type**.

$\mathbb{I}$ as an MTraS-shape is better behaved on $\bowtie$:

Twisted Cube Category $\bowtie$ [PK19]

(Roughly) the subcategory of FinLinOrd (or Δ) generated by $\top$ and $\sqcup \ltimes \mathbb{I}$.

n-Fold category

An ***n*-fold category** $\mathscr{C}$ is an ***n*-fold twisted cubical set** $\mathscr{C} \in \mathrm{Psh}(\bowtie^n)$ ~~satisfying the **Segal condition** in each dimension~~.

(Pro-arrow) Equipments

(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 $\varphi^{\ddagger} : x \not\to y$ and $\varphi^{\dagger} : y \not\to x$ 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 $R(a, b) \Rightarrow S(f\,a, g\,b)$

(Pro-arrow) Equipments

(Pro-arrow) Equipment

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

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

such that every arrow $\varphi : x \to y$ has **graph** pro-arrows $\varphi^{\ddagger} : x \nrightarrow y$ and $\varphi^{\dagger} : y \nrightarrow x$ 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 $R(a, b) \Rightarrow S(f\,a, g\,b)$

(Pro-arrow) Equipments

(Pro-arrow) Equipment

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

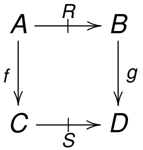
- objects
- arrows ($\to$)
- pro-arrows ($\nrightarrow$)
- squares

such that every arrow $\varphi : x \to y$ has **graph** pro-arrows $\varphi^{\ddagger} : x \nrightarrow y$ and $\varphi^{\dagger} : y \nrightarrow x$ 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 $R(a, b) \Rightarrow S(f\,a, g\,b)$



(Pro-arrow) Equipment

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

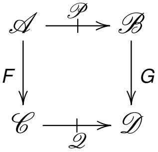
- objects
- arrows ($\rightarrow$)
- pro-arrows ($\nrightarrow$)
- squares

such that every arrow $\varphi : x \rightarrow y$ has **graph** pro-arrows $\varphi^{\ddagger} : x \nrightarrow y$ and $\varphi^{\dagger} : y \nrightarrow x$ such that $(\ldots)$.

Example: Cat

Cat is an equipment with:

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



Higher 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** (. . .)

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

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** (. . .)

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

Higher 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** (. . .)

$\Rightarrow \mathscr{C} \in \mathrm{Psh}(\bowtie^{n}_{\dagger,\ddagger})$

Higher 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** (. . .)

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

Higher 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

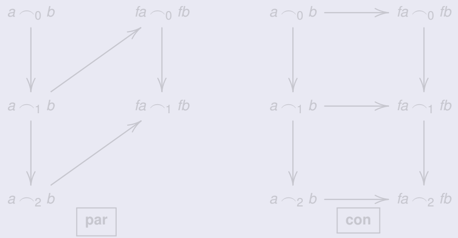
An n-**equipment** is an n-**fold category** (. . .)

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

Directifying “Degrees of Relatedness” [ND18]

Depth n types

- i-edge relations $\frown_i$
- $R : A \frown^{\mathrm{U}}_{i+1} B$
  is a container for $r : a \frown^{R}_{i} b$
  $\mathrm{U} = \langle \textbf{disc} \mid \mathrm{U}^{\mathrm{HS}} \rangle$
- $a \frown_i b \Rightarrow a \frown_{i+1} b$
- Modalities change indices:



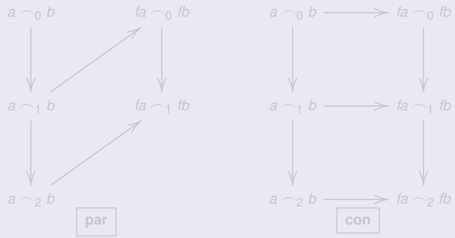
n-equipments

- i-jet (pro^{i-1}-arrow) relations $\curvearrowright_i$
- $J : A \curvearrowright^{\mathrm{U}}_{i+1} B$
  is a container for $j : a \curvearrowright^{J}_{i} b$
  $\mathrm{U} = \langle \textbf{disc} \mid \mathrm{U}^{\mathrm{HS}} \rangle$
- $(\ddagger, \dagger) : a \curvearrowright_i b \Rightarrow a \tilde{\curvearrowright}_{i+1} b$
- Modalities change indices & orientation:

Directifying "Degrees of Relatedness" [ND18]

Depth n types

- i-edge relations $\frown_i$
- $R : A \frown^{\mathrm{U}}_{i+1} B$
  is a container for $r : a \frown^{R}_{i} b$
  $\mathrm{U} = \langle \textbf{disc} \mid \mathrm{U}^{\mathrm{HS}} \rangle$
- $a \frown_i b \Rightarrow a \frown_{i+1} b$
- Modalities change indices:



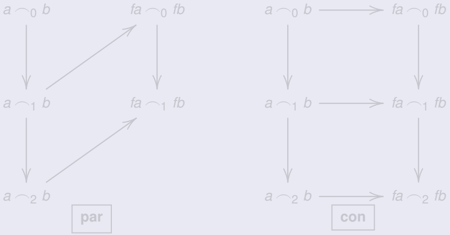
n-equipments

- i-jet (pro^{i-1}-arrow) relations $\frown_i$
- $J : A \frown^{\mathrm{U}}_{i+1} B$
  is a container for $j : a \frown^{J}_{i} b$
  $\mathrm{U} = \langle \mathrm{disc} \mid \mathrm{U}^{\mathrm{HS}} \rangle$
- $(\ddagger, \dagger) : a \frown_i b \Rightarrow a \tilde{\frown}_{i+1} b$
- Modalities change indices & orientation:

Directifying “Degrees of Relatedness” [ND18]

Depth n types

- i-edge relations $\frown_i$
- $R : A \frown^{\mathrm{U}}_{i+1} B$
  is a container for $r : a \frown^{R}_{i} b$
  $\mathrm{U} = \langle \textbf{disc} \mid \mathrm{U}^{\mathrm{HS}} \rangle$
- $a \frown_i b \Rightarrow a \frown_{i+1} b$
- Modalities change indices:



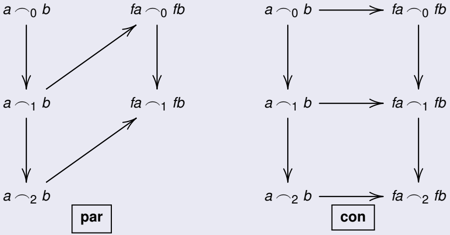
n-equipments

- i-jet (pro^{i-1}-arrow) relations $\rightsquigarrow_i$
- $J : A \rightsquigarrow^{\mathrm{U}}_{i+1} B$
  is a container for $j : a \rightsquigarrow^{J}_{i} b$
  $\mathrm{U} = \langle \mathrm{disc} \mid \mathrm{U}^{\mathrm{HS}} \rangle$
- $(\ddagger, \dagger) : a \rightsquigarrow_i b \Rightarrow a \tilde{\rightsquigarrow}_{i+1} b$
- Modalities change indices & orientation:

Directifying "Degrees of Relatedness" [ND18]

Depth n types

- i-edge relations $\frown_i$
- $R : A \frown^{\mathrm{U}}_{i+1} B$
  is a container for $r : a \frown^{R}_{i} b$
  $\mathrm{U} = \langle$ **disc** $\mid \mathrm{U}^{\mathrm{HS}} \rangle$
- $a \frown_i b \Rightarrow a \frown_{i+1} b$
- Modalities change indices:



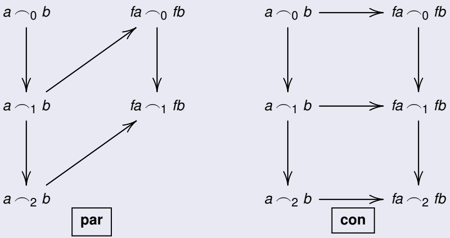
n-equipments

- i-jet (pro^{i-1}-arrow) relations $\frown_i$
- $J : A \frown^{\mathrm{U}}_{i+1} B$
  is a container for $j : a \frown^{J}_{i} b$
  $\mathrm{U} = \langle \mathrm{disc} \mid \mathrm{U}^{\mathrm{HS}} \rangle$
- $(\ddagger, \dagger) : a \frown_i b \Rightarrow a \tilde{\frown}_{i+1} b$
- Modalities change indices & orientation:

Directifying "Degrees of Relatedness" [ND18]

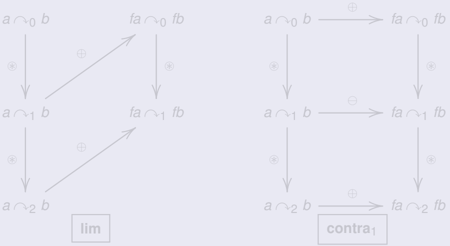
Depth n types

- i-edge relations $\frown_i$
- $R : A \frown^{\mathrm{U}}_{i+1} B$
  is a container for $r : a \frown^R_i b$
  $\mathrm{U} = \langle \textbf{disc} \mid \mathrm{U}^{\mathrm{HS}} \rangle$
- $a \frown_i b \Rightarrow a \frown_{i+1} b$
- Modalities change indices:



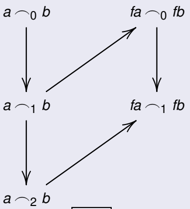
n-equipments

- i-jet (pro^{i-1}-arrow) relations $\curvearrowright_i$
- $J : A \curvearrowright^{\mathrm{U}}_{i+1} B$
  is a container for $j : a \curvearrowright^J_i b$
  $\mathrm{U} = \langle \textbf{disc} \mid \mathrm{U}^{\mathrm{HS}} \rangle$
- $(\ddagger, \dagger) : a \curvearrowright_i b \Rightarrow a \,\tilde{\curvearrowright}_{i+1}\, b$
- Modalities change indices & orientation:

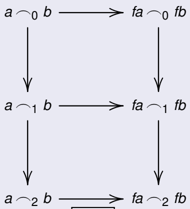


Depth n types

- i-edge relations $\frown_i$
- $R : A \frown^{\mathsf{U}}_{i+1} B$
  is a container for $r : a \frown^R_i b$
  $\mathsf{U} = \langle \textbf{disc} \mid \mathsf{U}^{\mathrm{HS}} \rangle$
- $a \frown_i b \Rightarrow a \frown_{i+1} b$
- Modalities change indices:



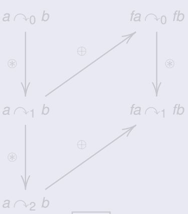
par



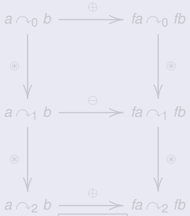
con

n-equipments

- i-jet (pro^{i-1}-arrow) relations $\curvearrowright_i$
- $J : A \curvearrowright^{\mathsf{U}}_{i+1} B$
  is a container for $j : a \curvearrowright^J_i b$
  $\mathsf{U} = \langle \textbf{disc} \mid \mathsf{U}^{\mathrm{HS}} \rangle$
- $(\ddagger, \dagger) : a \curvearrowright_i b \Rightarrow a \tilde{\curvearrowright}_{i+1} b$
- Modalities change indices & orientation:



lim

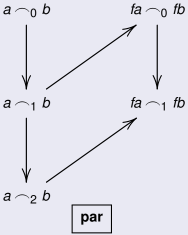


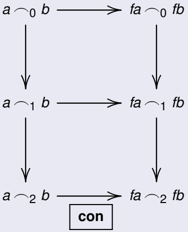
contra$_1$

Directifying "Degrees of Relatedness" [ND18]

Depth n types

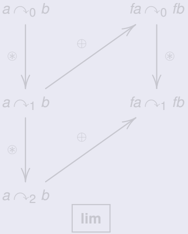
- i-edge relations $\frown_i$
- $R : A \frown^{\mathrm{U}}_{i+1} B$
  is a container for $r : a \frown^{R}_{i} b$
  $\mathrm{U} = \langle \textbf{disc} \mid \mathrm{U}^{\mathrm{HS}} \rangle$
- $a \frown_i b \Rightarrow a \frown_{i+1} b$
- Modalities change indices:

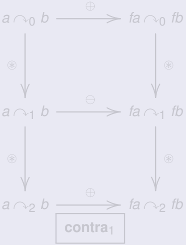




n-equipments

- i-jet (pro^{i-1}-arrow) relations $\curvearrowright_i$
- $J : A \curvearrowright^{\mathrm{U}}_{i+1} B$
  is a container for $j : a \curvearrowright^{J}_{i} b$
  $\mathrm{U} = \langle \textbf{disc} \mid \mathrm{U}^{\mathrm{HS}} \rangle$
- $(\ddagger, \dagger) : a \curvearrowright_i b \Rightarrow a \,\tilde{\curvearrowright}_{i+1}\, b$
- Modalities change indices & orientation:

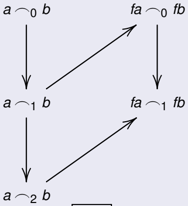




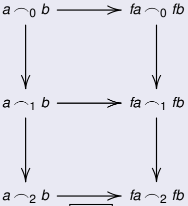
Directifying “Degrees of Relatedness” [ND18]

Depth n types

- i-edge relations $\frown_i$
- $R : A \frown^{\mathrm{U}}_{i+1} B$
  is a container for $r : a \frown^{R}_{i} b$
  $\mathrm{U} = \langle$ **disc** $\mid \mathrm{U}^{\mathrm{HS}} \rangle$
- $a \frown_i b \Rightarrow a \frown_{i+1} b$
- Modalities change indices:



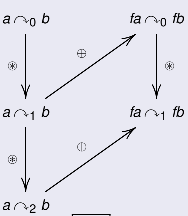
par



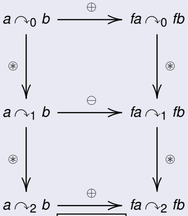
con

n-equipments

- i-jet (pro^{i-1}-arrow) relations $\curvearrowright_i$
- $J : A \curvearrowright^{\mathrm{U}}_{i+1} B$
  is a container for $j : a \curvearrowright^{J}_{i} b$
  $\mathrm{U} = \langle$ **disc** $\mid \mathrm{U}^{\mathrm{HS}} \rangle$
- $(\ddagger, \dagger) : a \curvearrowright_i b \Rightarrow a \,\tilde{\curvearrowright}_{i+1}\, b$
- Modalities change indices & orientation:



lim



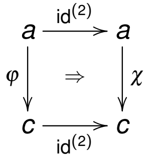
contra$_1$

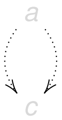
2-category (Tamsamani & Simpson)

An 2-category is an **double (2-fold)** category whose (2)-arrows are all trivial (id), so only

- (1)-arrows ⇝ 1-arrows
- (1,2)-squares ⇝ 2-arrows

can be non-trivial.





n-category (Tamsamani & Simpson)

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

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

can be non-trivial.

Recall the equipment Cat:

$$\overset{\text{end}}{\forall}\, a, b. \mathscr{P}(a, b) \Rightarrow \mathscr{Q}(F a, G b)$$
$$\cong \overset{\text{end}}{\forall}\, a. \mathrm{Hom}(F a, G a)$$

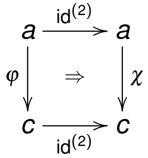
Tamsamani & Simpson's model of higher category theory

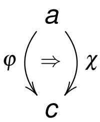
2-category (Tamsamani & Simpson)

An 2-category is an **double (2-fold)** category whose (2)-arrows are all trivial (id), so only

- (1)-arrows $\rightsquigarrow$ 1-arrows
- (1,2)-squares $\rightsquigarrow$ 2-arrows

can be non-trivial.





n-category (Tamsamani & Simpson)

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

- (1)-arrows, $\rightsquigarrow$ 1-arrows
- (1,2)-squares, $\rightsquigarrow$ 2-arrows
- (1,2,3)-cubes, $\rightsquigarrow$ 3-arrows …

can be non-trivial.

Recall the equipment Cat:

$$\overset{\text{end}}{\forall}\, a, b. \mathscr{P}(a, b) \Rightarrow \mathscr{Q}(F\,a, G\,b)$$

$$\cong \overset{\text{end}}{\forall}\, a. \mathrm{Hom}(F\,a, G\,a)$$

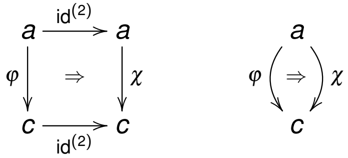
Tamsamani & Simpson's model of higher category theory

2-category (Tamsamani & Simpson)

An 2-category is an **double (2-fold)** category whose (2)-arrows are all trivial (id), so only

- (1)-arrows $\rightsquigarrow$ 1-arrows
- (1,2)-squares $\rightsquigarrow$ 2-arrows

can be non-trivial.



Recall the equipment Cat:

$$\overset{\text{end}}{\forall}\, a, b. \mathscr{P}(a,b) \Rightarrow \mathscr{Q}(F\,a, G\,b)$$
$$\cong \quad \overset{\text{end}}{\forall}\, a. \mathrm{Hom}(F\,a, G\,a)$$

n-category (Tamsamani & Simpson)

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

- (1)-arrows, $\rightsquigarrow$ 1-arrows
- (1,2)-squares, $\rightsquigarrow$ 2-arrows
- (1,2,3)-cubes, $\rightsquigarrow$ 3-arrows ...

can be non-trivial.

Tamsamani & Simpson's model of higher category theory

2-category (Tamsamani & Simpson)

An 2-category is an **double (2-fold)** category whose (2)-arrows are all trivial (id), so only

- (1)-arrows $\rightsquigarrow$ 1-arrows
- (1,2)-squares $\rightsquigarrow$ 2-arrows

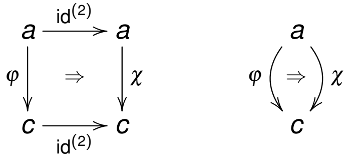
can be non-trivial.

n-category (Tamsamani & Simpson)

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

- (1)-arrows, $\rightsquigarrow$ 1-arrows
- (1,2)-squares, $\rightsquigarrow$ 2-arrows
- (1,2,3)-cubes, $\rightsquigarrow$ 3-arrows . . .

can be non-trivial.



Recall the equipment Cat:

$$\overset{\text{end}}{\forall}\, a, b. \mathscr{P}(a, b) \Rightarrow \mathscr{Q}(F a, G b)$$
$$\cong \quad \overset{\text{end}}{\forall}\, a. \mathrm{Hom}(F a, G a)$$

2-category (Tamsamani & Simpson)

An 2-category is an **double (2-fold)** category whose (2)-arrows are all trivial (id), so only

- (1)-arrows $\rightsquigarrow$ 1-arrows
- (1,2)-squares $\rightsquigarrow$ 2-arrows

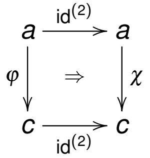
can be non-trivial.

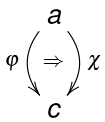
n-category (Tamsamani & Simpson)

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

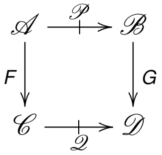
- (1)-arrows, $\rightsquigarrow$ 1-arrows
- (1,2)-squares, $\rightsquigarrow$ 2-arrows
- (1,2,3)-cubes, $\rightsquigarrow$ 3-arrows ...

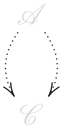
can be non-trivial.





Recall the equipment Cat:





$$\overset{\text{end}}{\forall}\, a, b.\mathscr{P}(a,b) \Rightarrow \mathscr{Q}(F\,a, G\,b)$$

$$\cong \overset{\text{end}}{\forall}\, a.\mathrm{Hom}(F\,a, G\,a)$$

Tamsamani & Simpson's model of higher category theory

2-category (Tamsamani & Simpson)

An 2-category is an **double (2-fold)** category whose (2)-arrows are all trivial (id), so only

- (1)-arrows $\rightsquigarrow$ 1-arrows
- (1,2)-squares $\rightsquigarrow$ 2-arrows

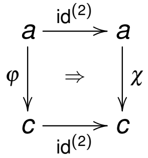
can be non-trivial.

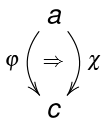
n-category (Tamsamani & Simpson)

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

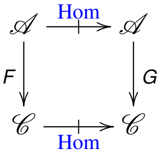
- (1)-arrows, $\rightsquigarrow$ 1-arrows
- (1,2)-squares, $\rightsquigarrow$ 2-arrows
- (1,2,3)-cubes, $\rightsquigarrow$ 3-arrows ...

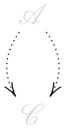
can be non-trivial.





Recall the equipment Cat:





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

$$\cong \quad \overset{\text{end}}{\forall}\, a. \mathrm{Hom}(F\,a, G\,a)$$

2-category (Tamsamani & Simpson)

An 2-category is an **double (2-fold)** category whose (2)-arrows are all trivial (id), so only

- (1)-arrows $\rightsquigarrow$ 1-arrows
- (1,2)-squares $\rightsquigarrow$ 2-arrows

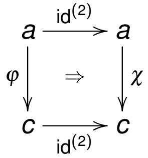
can be non-trivial.

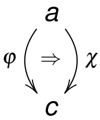
n-category (Tamsamani & Simpson)

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

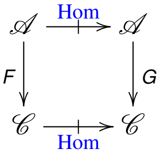
- (1)-arrows, $\rightsquigarrow$ 1-arrows
- (1,2)-squares, $\rightsquigarrow$ 2-arrows
- (1,2,3)-cubes, $\rightsquigarrow$ 3-arrows ...

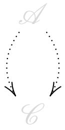
can be non-trivial.





Recall the equipment Cat:





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

$$\cong \quad \overset{\text{end}}{\forall}\, a.\mathrm{Hom}(F\,a, G\,a)$$

Tamsamani & Simpson's model of higher category theory

2-category (Tamsamani & Simpson)

An 2-category is an **double (2-fold)** category whose (2)-arrows are all trivial (id), so only

- (1)-arrows $\rightsquigarrow$ 1-arrows
- (1,2)-squares $\rightsquigarrow$ 2-arrows

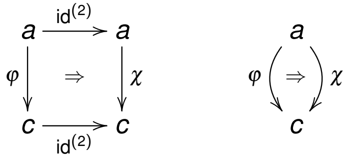
can be non-trivial.

n-category (Tamsamani & Simpson)

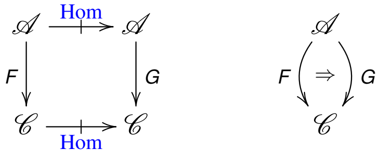
An *n*-category is an ***n*-fold** category where only

- (1)-arrows, $\rightsquigarrow$ 1-arrows
- (1,2)-squares, $\rightsquigarrow$ 2-arrows
- (1,2,3)-cubes, $\rightsquigarrow$ 3-arrows ...

can be non-trivial.



Recall the equipment Cat:



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

$$\cong \quad \overset{\text{end}}{\forall}\, a.\mathrm{Hom}(F\,a, G\,a)$$

Tamsamani & Simpson's model of higher category theory

2-category (Tamsamani & Simpson)

An 2-category is an **double (2-fold)** category whose (2)-arrows are all trivial (id), so only

- (1)-arrows $\rightsquigarrow$ 1-arrows
- (1,2)-squares $\rightsquigarrow$ 2-arrows

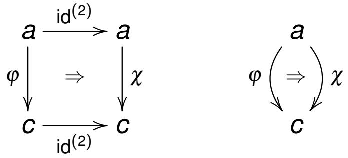
can be non-trivial.

n-category (Tamsamani & Simpson)

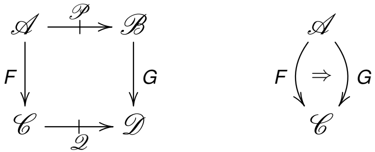
An *n*-category is an ***n*-fold** category where only

- (1)-arrows, $\rightsquigarrow$ 1-arrows
- (1,2)-squares, $\rightsquigarrow$ 2-arrows
- (1,2,3)-cubes, $\rightsquigarrow$ 3-arrows …

can be non-trivial.



Recall the equipment Cat:



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

$$\cong \overset{\text{end}}{\forall}\, a.\mathrm{Hom}(F\,a, G\,a)$$

Status of the model

- The building blocks are there (also further ahead!).
- Sort out details of base category & modalities. (Not success/failure but descriptive.)

Conclusion

We are **not** stuck on *higher* directed type theory.

Thanks!

Questions?

Status of the model

- The building blocks are there (also further ahead!).
- Sort out details of base category & modalities. (Not success/failure but descriptive.)

Conclusion

We are **not** stuck on *higher* directed type theory.

Thanks!

Questions?