# AdapTT: A Type Theory with Functorial Types

**Arthur Adjedj**[1,2], Thibaut Benjamin[2]
Meven Lennon-Bertrand[2], Kenji Maillard[3]

[1]ENS Paris Saclay, Gif-sur-Yvette, France
[2]University of Cambridge, Cambridge, United Kingdom
[3]Gallinette Project Team, Inria, Nantes, France

Observational equality

Definitional equality

# Type casts are everywhere

Gradual Types

Subtyping

# A few examples

| | | | |
|---|---|---|---|
| Gradual Types (CastCIC[1]) | $\mathrm{cast}_{\Pi(x:A_1).B_1 \atop \Pi(x:A_2).B_2}(f)$ | $\Rightarrow$ | $\lambda\ (a_2 : A_2) \Rightarrow$ let $a_1 := \mathrm{cast}_{A_2,A_1}(a_2)$ in $\mathrm{cast}_{B_1[a_1/x],B_2[a_2/x]}(f\ a_1)$ |
| | | $\Rightarrow$ | |
| | | | |

# A few examples

| | | | |
|---|---|---|---|
| Gradual Types (CastCIC[1]) | $\text{cast}_{\Pi(x:A_1).B_1 \atop \Pi(x:A_2).B_2}(f)$ | $\Rightarrow$ | $\lambda\ (a_2 : A_2) \Rightarrow$ <br> $\text{let } a_1 := \text{cast}_{A_2,A_1}(a_2) \text{ in}$ <br> $\text{cast}_{B_1[a_1/x],B_2[a_2/x]}(f\ a_1)$ |
| Observational Equality ($\text{TT}^{\text{obs}}$[2]) | $\text{cast}_{\Pi(x:A_1).B_1 \atop \Pi(x:A_2).B_2}(e,\ f)$ | $\Rightarrow$ | $\lambda\ (a_2 : A_2) \Rightarrow$ <br> $\text{let } a_1 := \text{cast}_{A_2,A_1}(\text{fst}(e)^{-1}, a_2) \text{ in}$ <br> $\text{cast}_{B_1[a_1/x],B_2[a_2/x]}(\text{snd}(e)\ a_2, f\ a_1)$ |
| | | $\Rightarrow$ | |

# A few examples

| | | |
|---|---|---|
| Gradual Types (CastCIC[1]) | $\mathrm{cast}_{\Pi(x:A_1).B_1}^{\Pi(x:A_2).B_2}(f) \Rightarrow$ | $\lambda\ (a_2 : A_2) \Rightarrow$ let $a_1 := \mathrm{cast}_{A_2,A_1}(a_2)$ in $\mathrm{cast}_{B_1[a_1/x],B_2[a_2/x]}(f\ a_1)$ |
| Observational Equality ($\mathrm{TT}^{\mathrm{obs}}$[2]) | $\mathrm{cast}_{\Pi(x:A_1).B_1}^{\Pi(x:A_2).B_2}(e,\ f) \Rightarrow$ | $\lambda\ (a_2 : A_2) \Rightarrow$ let $a_1 := \mathrm{cast}_{A_2,A_1}(\mathrm{fst}(e)^{-1}, a_2)$ in $\mathrm{cast}_{B_1[a_1/x],B_2[a_2/x]}(\mathrm{snd}(e)\ a_2, f\ a_1)$ |
| Coercive Subtyping ($\mathrm{MLTT}_{\mathrm{coe}}$[3]) | $\mathrm{coe}_{\Pi x:A_1.B_1}^{\Pi x:A_2.B_2}(f) \Rightarrow$ | $\lambda\ (a_2 : A_2) \Rightarrow$ let $a_1 := \mathrm{coe}_{A_2,A_1}\ a_2$ in $\mathrm{coe}_{B_1[a_1/x],B_2[a_2/x]}\ (f\ a_1)$ |

# A few examples

$(a_2 : A_2) \Rightarrow$

$\quad \text{let } a_1 := \text{cast}_{A_2,A_1}(a_2) \text{ in}$

$\quad \text{cast}_{B_1[a_1/x],B_2[a_2/x]}(f\ a_1)$

$(a_2 : A_2) \Rightarrow$

$\text{let } a_1 := \text{cast}_{A_2,A_1}(\text{fst}(e)^{-1}, a_2) \text{ in}$

$\text{cast}_{B_1[a_1/x],B_2[a_2/x]}(\text{snd}(e)\ a_2, f\ a_1)$

$(a_2 : A_2) \Rightarrow$

$\quad \text{let } a_1 := \text{coe}_{A_2,A_1}\ a_2 \text{ in}$

$\quad \text{coe}_{B_1[a_1/x],B_2[a_2/x]}\ (f\ a_1)$

# A common core

Exponential in a Cartesian Closed Category:

$$\begin{aligned} \mathbb{C}^{\text{op}} \times \mathbb{C} &\to \mathbb{C} \\ (A, B) &\mapsto B^A \\ (f, g) &\mapsto h \mapsto x \mapsto g\ (\text{eval}\ (h, f(x))) \end{aligned}$$

# A common core

Exponential in a Cartesian Closed Category:

$$\begin{aligned} \mathbb{C}^{\mathrm{op}} \times \mathbb{C} &\to \mathbb{C} \\ (A, B) &\mapsto B^A \\ (f, g) &\mapsto h \mapsto x \mapsto g\ (\mathrm{eval}\ (h, f(x))) \end{aligned}$$

**These theories exhibit functorial properties of Π !**

This functorial property acts over many forms of type casts (propositional equality, definitional equality, subtyping)

# Objectives

- Construct a framework to describe type casts over $\Pi$ using its functorial property
- Other type formers exist (Id, $\Sigma$, W, ...), and user can create new ones: Inductive types
  Casts should compute over arbitrary inductive types.
  How to exhibit their functoriality ?

# Functors ? in my Set ?

Where do we start from ? Categories with Families (CwF)

The big picture:

- A **category** $\mathrm{Ctx}$ of contexts, morphisms are substitutions
- A functor $T : \mathrm{Ctx}^{\mathrm{op}} \to \mathbf{Fam}$ i.e:
  - $\mathrm{Ty} : \mathrm{Ctx}^{\mathrm{op}} \to \mathbf{Set}$
  - $\mathrm{Tm} : \int_{\mathrm{Ctx}^{\mathrm{op}}} \mathrm{Ty} \to \mathbf{Set}$
- Variables, context extensions, ...

**What should we change to have functors between types ?**

# Functors ? in my Set ?

Types now form a category:

- A **category** $\mathrm{Ctx}$ of contexts, morphisms are substitutions
- A functor $T : \mathrm{Ctx}^{\mathrm{op}} \to$ **Cat** ⫽ **Set** i.e:
  - $\mathrm{Ty} : \mathrm{Ctx}^{\mathrm{op}} \to$ **Cat**
  - $\mathrm{Tm} : \int_{\mathrm{Ctx}^{\mathrm{op}}} \mathrm{Ty} \to$ **Set**

**Adapters**[4]:

$$\mathrm{Ad}(\Gamma, A, B) = \mathrm{Hom}_{\mathrm{Ty}(\Gamma)}(A, B)$$

$$\frac{\Gamma : \mathrm{Ctx} \quad A, B : \mathrm{Ty}(\Gamma) \quad t : A \qquad a : \mathrm{Ad}(\Gamma, A, B)}{t\langle a \rangle : \mathrm{Tm}(\Gamma, B)}$$

# Type formers as natural transformations

Data of a type former :

- A presheaf $D : \mathrm{Ctx}^{\mathrm{op}} \to \mathbf{Cat}$ = "input data"
- A natural transformation $C : D \Rightarrow \mathsf{Ty}$
  - On objects: the type former
  - On morphisms: structural coercion

# Type formers as natural transformations

Data of a type former :

- A presheaf $D : \mathrm{Ctx}^{\mathrm{op}} \to \mathbf{Cat}$ = "input data"
- A natural transformation $C : D \Rightarrow \mathsf{Ty}$
  - On objects: the type former
  - On morphisms: structural coercion

Examples :

| Type Constructor | List | $\Pi$ |
|---|---|---|
| Presheaf $D(\Gamma)$ | $\mathrm{Ty}(\Gamma)$ | $(A : \mathsf{Ty}^{\mathrm{op}}(-)) \times \mathrm{Ty}(- \triangleright A))$ |
| $C(\Gamma)$ on objects | $A \mapsto \mathrm{List}_{\Gamma}(A)$ | $(A, B) \mapsto \Pi A.B$ |
| $C(\Gamma)$ on morphisms | $(f, t) \mapsto \mathrm{map}\ f\ t$ | $((f, g), t) \mapsto \lambda\ (a_2 : A_2) \Rightarrow$ let $a_1 := f(a_2)$ in $g(a_2, t\ a_1)$ |

# Type formers as natural transform

Data of a type former :

- A presheaf $D : \mathrm{Ctx}^{\mathrm{op}} \to \mathbf{Cat}$ = input data
- A natural transformation $C : D \Rightarrow \mathsf{Ty}$
  - On objects = the applied type former.
  - On morphisms = a coercion between instances of the type former.

Very general ! *Too* general...

We want:

- A syntactic presentation of type formers
- That explicits the **variance** data
- Powerful enough to encode usual type formers ($\Pi$, $\Sigma$, $\mathrm{Id}$, $\mathrm{W}$)

# 1-Yoneda to the rescue ?

Data of a type former :

- A presheaf $D : \mathrm{Ctx}^{\mathrm{op}} \to \mathbf{Cat}$ = input data
- A natural transformation $C : D \Rightarrow$ Ty
  - On objects = the applied type former.
  - On morphisms = a coercion between instances of the type former.

**Theorem :** $\mathrm{Ty}(\Gamma)$ is in bijection with $\mathrm{Sub}(-, \Gamma) \Rightarrow \mathrm{Ob} \circ \mathrm{Ty}$.

As such, any $F : \mathrm{Ty}(\Gamma)$ gives rise to a type-former !

# 1-Yoneda to the rescue ?

Data of a type former :

- A presheaf $D : \mathrm{Ctx}^{\mathrm{op}} \to \mathbf{Cat}$ = input data
- A natural transformation $C : D \Rightarrow \mathsf{Ty}$
  - On objects = the applied type former.
  - On morphisms = a coercion between instances of the type former.

**Theorem :** $\mathrm{Ty}(\Gamma)$ is in bijection with $\mathrm{Sub}(-, \Gamma) \Rightarrow \mathrm{Ob} \circ \mathrm{Ty}$.

As such, any $F : \mathrm{Ty}(\Gamma)$ gives rise to a type-former !

Nice, but…

- Can't capture interesting examples ($\Pi$, $\Sigma$, …)
- Just a bijection, what about our **adapters** ?

# Type variables

What we want for $\Pi$:

$$\Gamma_{\Pi} := (X : \mathsf{Ty}^{-}) \triangleright (Y : (X.\mathsf{Ty}^{+}))$$

# Type variables

What we want for $\Pi$:

$$\Gamma_\Pi := (X : \mathsf{Ty}^-) \triangleright (Y : (X.\mathsf{Ty}^+))$$

A type variable in $\Gamma$:

- Binds a telescope $\Theta : \mathrm{Tel}(\Gamma)$
- Has a direction

# Type variables

What we want for $\Pi$:

$$\Gamma_{\Pi} := (X : \mathsf{Ty}^{-}) \triangleright (Y : (X.\mathsf{Ty}^{+}))$$

A type variable in $\Gamma$:

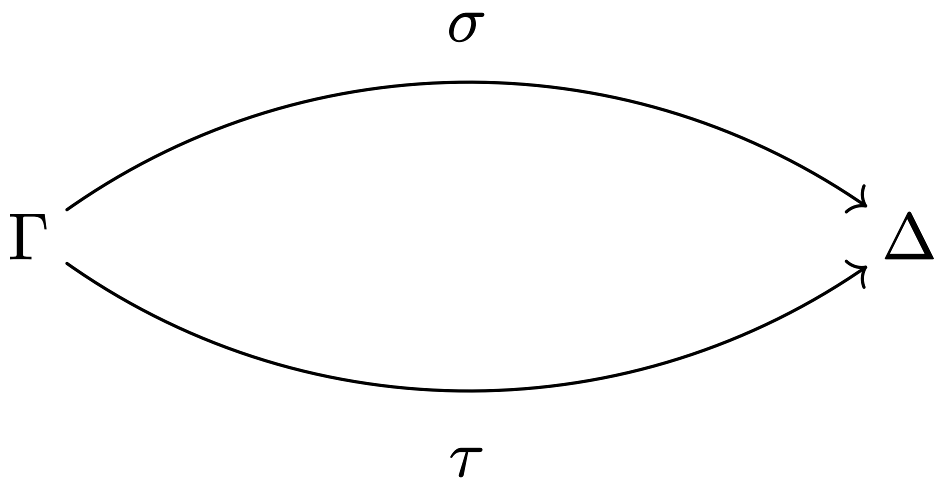
- Binds a telescope $\Theta : \mathrm{Tel}(\Gamma)$
- Has a direction

Works great for others:

$$\mathrm{Id} : \Gamma_{\mathrm{Id}} := (X : \mathsf{Ty}^{+}) \triangleright X \qquad \Sigma : \Gamma_{\Sigma} := (X : \mathsf{Ty}^{+}) \triangleright (Y : (X.\mathsf{Ty}^{+}))$$

# Contexts as 2-categorical objects

Substitutions map type variables to types.

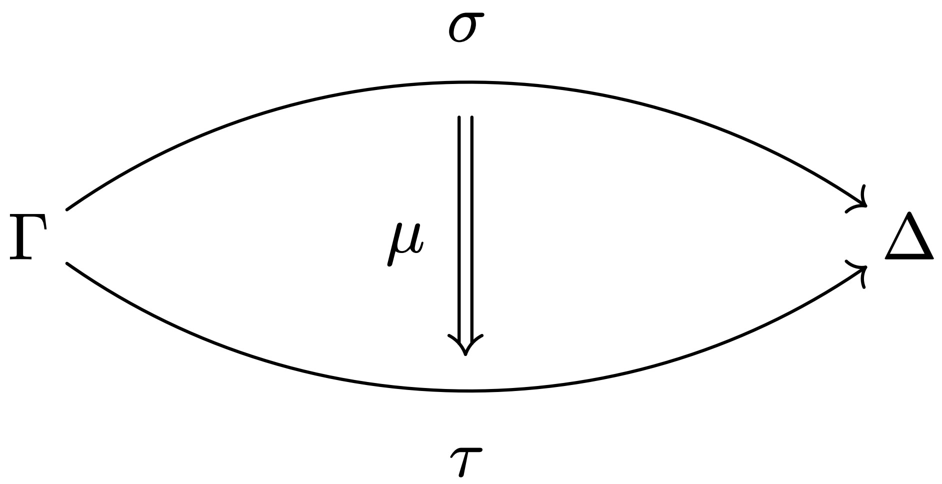


# Contexts as 2-categorical objects

Substitutions map type variables to types.

Types are related through **adapters** collected into **transformations**.

Ctx becomes a **2-Category**.




$$\frac{\Gamma, \Delta : \mathrm{Ctx} \quad \sigma, \tau : \mathrm{Sub}(\Gamma, \Delta)}{\mathrm{Trans}(\Gamma, \Delta, \delta, \tau)}$$

# 2-Yoneda is useful

Data of a type former :

- A presheaf $D : \mathrm{Ctx}^{\mathrm{op}} \to \mathbf{Cat}$ = input data
- A natural transformation $C : D \Rightarrow \mathsf{Ty}$
  - On objects = the applied type former.
  - On morphisms = a coercion between instances of the type former.

**Theorem :** $\mathrm{Ty}(\Gamma)$ is **isomorphic** to $\mathrm{Sub}(-, \Gamma) \Rightarrow \mathrm{Ty}$.

# 2-Yoneda is useful

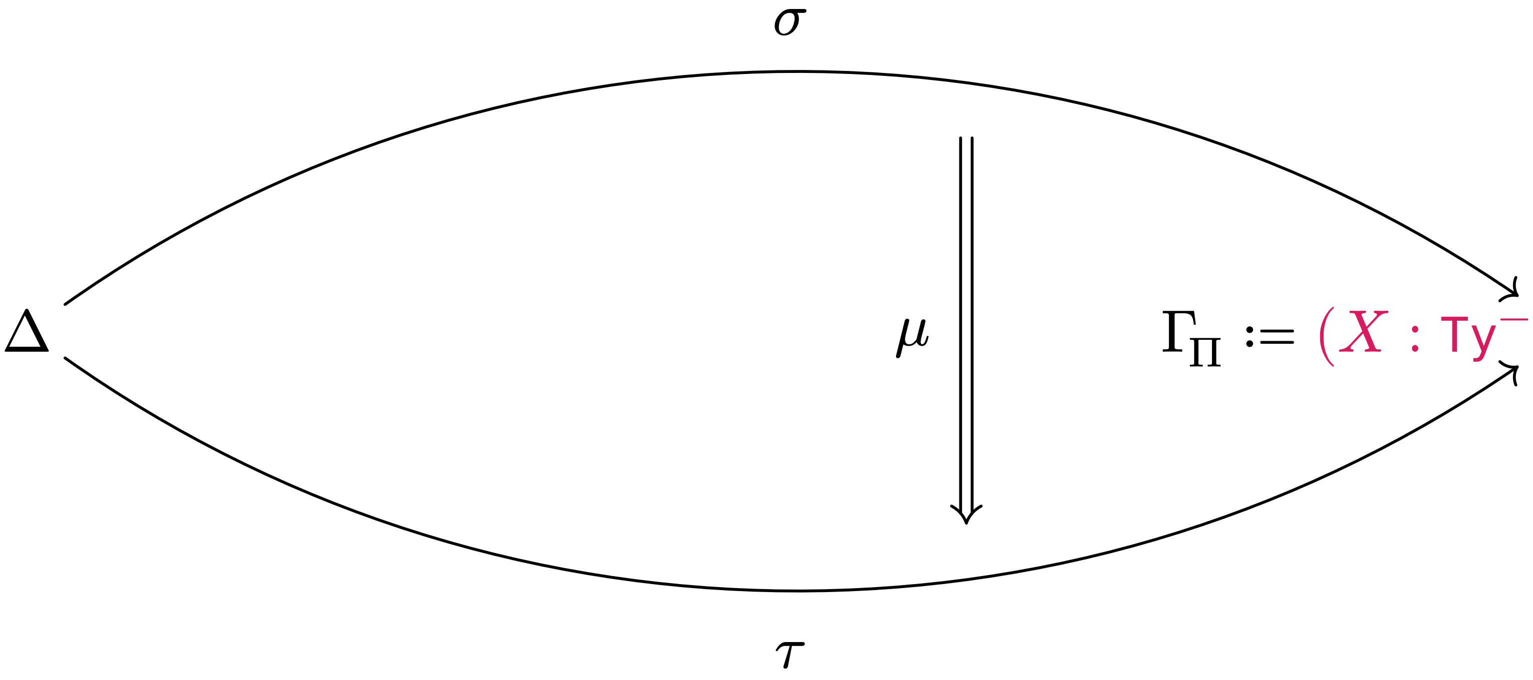
Data of a type former :

- A presheaf $D : \mathrm{Ctx}^{\mathrm{op}} \to \mathbf{Cat}$ = input data
- A natural transformation $C : D \Rightarrow$ Ty
  - On objects = the applied type former.
  - On morphisms = a coercion between instances of the type former.

**Theorem :** $\mathrm{Ty}(\Gamma)$ is **isomorphic** to $\mathrm{Sub}(-,\Gamma) \Rightarrow \mathrm{Ty}$.

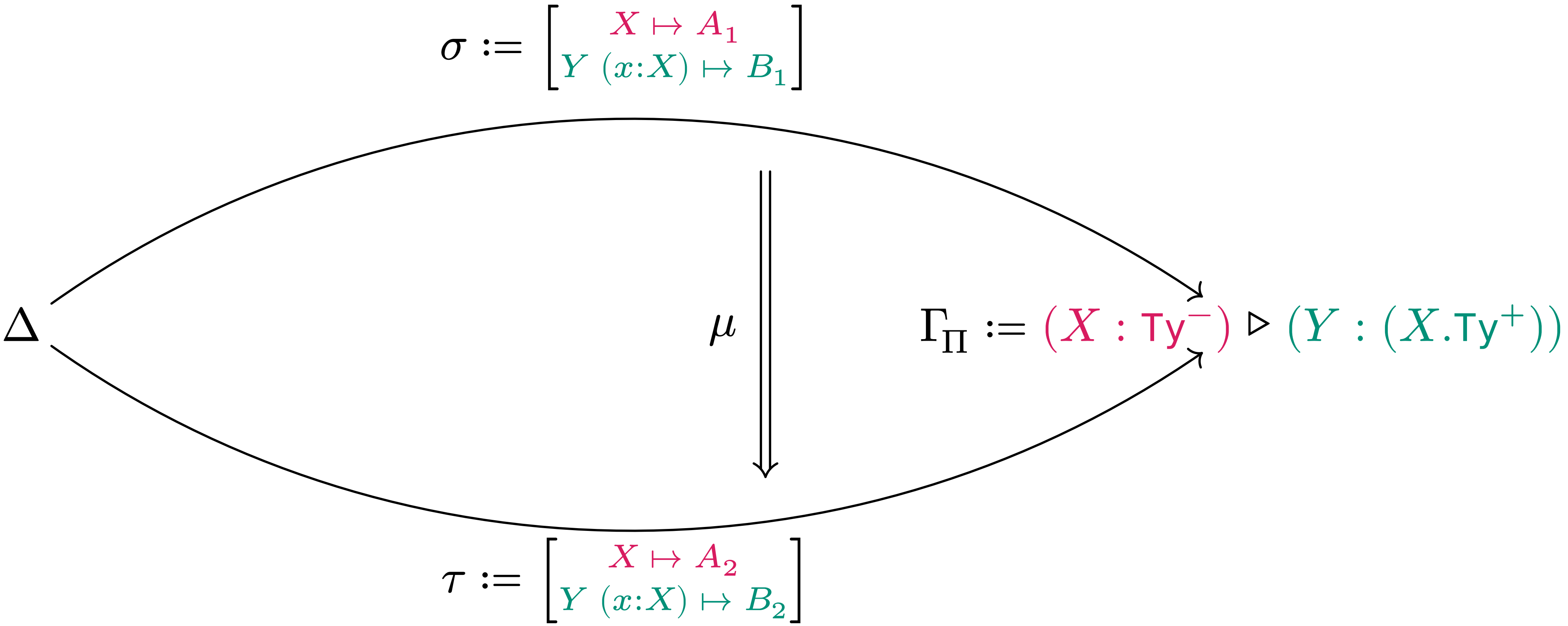
As such, any $F : \mathrm{Ty}(\Gamma)$ gives rise to a **functorial** type-former !

# Example: Π




$$\Gamma_\Pi := (X : \mathsf{Ty}^-) \triangleright (Y : (X.\mathsf{Ty}^+))$$

# Example: $\Pi$


$\sigma := \begin{bmatrix} X \mapsto A_1 \\ Y\ (x{:}X) \mapsto B_1 \end{bmatrix}$
$\Delta$
$\mu$
$\Gamma_\Pi := (X : \mathsf{Ty}^-) \triangleright (Y : (X.\mathsf{Ty}^+))$
$\tau := \begin{bmatrix} X \mapsto A_2 \\ Y\ (x{:}X) \mapsto B_2 \end{bmatrix}$

# Example: Π

$$\sigma := \begin{bmatrix} X \mapsto A_1 \\ Y\ (x{:}X) \mapsto B_1 \end{bmatrix}$$

$$\Delta \qquad \mu := \begin{bmatrix} a_A \colon \mathrm{Ad}(\Delta, A_2, A_1) \\ a_B \colon \mathrm{Ad}(\Delta \triangleright A_1, B_1, B_2[\mathrm{id} \Rrightarrow a_A]) \end{bmatrix} \qquad \Gamma_\Pi := (X : \mathsf{Ty}^-) \triangleright (Y : (X.\mathsf{Ty}^+))$$

$$\tau := \begin{bmatrix} X \mapsto A_2 \\ Y\ (x{:}X) \mapsto B_2 \end{bmatrix}$$

# A quick summary

- A **category** $\mathrm{Ctx}$ of contexts, 1-cells are substitutions

- A **functor** $T : \mathrm{Ctx}^{\mathrm{op}} \to \mathbf{Fam}$ i.e:
  - $\mathrm{Ty} : \mathrm{Ctx}^{\mathrm{op}} \to \mathbf{Set}$
  - $\mathrm{Tm} : \int_{\mathrm{Ctx}^{\mathrm{op}}} \mathrm{Ty} \to \mathbf{Set}$
- Term variables, context extensions, …

# A quick summary

- A **2-category** $\mathrm{Ctx}$ of contexts, 1-cells are substitutions , 2-cells are pop[transformations]
- A **2-functor** $T : \mathrm{Ctx}^{\mathrm{op}} \to$ **Cat** ⫽ **Set** i.e:
  - $\mathrm{Ty} : \mathrm{Ctx}^{\mathrm{op}} \to \mathbf{Cat}$
  - $\mathrm{Tm} : \int_{\mathrm{Ctx}^{\mathrm{op}}} \mathrm{Ty} \to \mathbf{Set}$
- Term variables, **type variables,** context extensions, …

# A Theory of Signatures for Inductive Types ⚠️ WIP

What we want:

- Encode (non-mutual, non-nested) inductive types with parameters and indices
- Embed these types into our class of models
- Action of substitution and transformation over constructors

What we have:

- Contexts
- Type variables
- **Telescopes**

# A Theory of Signatures for Inductive Types ⚠️ WIP

A **simple** inductive type Ind is:

- A list $\vec{C}$ of **constructors**

A constructor is:

- A telescope $\Theta_{\text{norec}} : \text{Tel}_+(\varepsilon)$ of **non-recursive arguments**
- A list of **recursive arguments**

A recursive argument $(a_1 : A_1) \to \ldots \to (a_n : A_n) \to \text{Ind}$ is:

- A telescope $\Theta_{\text{rec}} := \varepsilon \triangleright A_1 \triangleright \ldots \triangleright A_n : \text{Tel}\_(\Gamma \triangleright \Theta_{\text{norec}})$ of **arity**

# A Theory of Signatures for Inductive Types ⚠️ WIP

A **parametrised** inductive type $\mathrm{Ind}$ is:

- A context $\Gamma$ of **parameters**
- A list $\vec{C}$ of **constructors**

A constructor is:

- A telescope $\Theta_{\mathrm{norec}} : \mathrm{Tel}(\Gamma)$ of **non-recursive arguments**
- A list of **recursive arguments**

A recursive argument $(a_1 : A_1) \to \ldots \to (a_n : A_n) \to \mathrm{Ind}\ \vec{P}$ is:

- A telescope $\Theta_{\mathrm{rec}} := \varepsilon \triangleright A_1 \triangleright \ldots \triangleright A_n : \mathrm{Tel\_}(\Gamma \triangleright \Theta_{\mathrm{norec}})$ of **arity**

# A Theory of Signatures for Inductive Types ⚠️ WIP

A **parametrised, indexed** inductive type Ind is:

- A context $\Gamma$ of **parameters**
- A telescope $\Theta_I : \mathrm{Tel}_+(\Gamma)$ of **indices**
- A list $\vec{C}$ of **constructors**

A constructor is:

- A telescope $\Theta_{\mathrm{norec}} : \mathrm{Tel}(\Gamma)$ of **non-recursive arguments**
- A list of **recursive arguments**
- An instantiation $\Theta_I$ of **indices** in $\Gamma \triangleright \Theta_{\mathrm{norec}}$

A recursive argument $(a_1 : A_1) \to \ldots \to (a_n : A_n) \to \mathrm{Ind}\ \vec{P}\ \vec{I}$ is:

- A telescope $\Theta_{\mathrm{rec}} := \varepsilon \triangleright A_1 \triangleright \ldots \triangleright A_n : \mathrm{Tel}\_(\Gamma \triangleright \Theta_{\mathrm{norec}})$ of **arity**
- An instantiation $\Theta_I$ of **indices** in $\Gamma \triangleright \Theta_{\mathrm{norec}} \triangleright \Theta_{\mathrm{rec}}$

# A Theory of Signatures for Inductive Types ⚠️ WIP

## Example : Bounded W-types

- Parameters :
  $\Gamma_{\text{Ind}} := (A : \mathtt{Ty}^{+}) \triangleright (B : (A.\mathtt{Ty}^{-})$
- Indices : $\Theta_I := (n : \mathbb{N})$
- Constructor:
  - Non-recursive fields:
    $\Theta_{\text{norec}} := (n : \mathbb{N}) \triangleright (a : A)$
  - Recursive field:
    - Telescope $\Theta_{\text{rec}} := (b : B\ a)$
    - Index instantiation : $n$
  - Index instantiation : $n + 1$

```
data BW (A : 𝓤) (B : A → 𝓤) : ℕ → 𝓤 where
  sup : (n : ℕ)
      → (a : A)
      → ((b : B a) → BW A B n)
      → BW A B (n+1)
```

# What's done so far

⚠️ WIP

- Type former ✓
- Constructors ✓
- Action of substitution ✓
- Action of transformations ✓
- Construction of the recursor ❌
- Fusion laws/recursors on adapters ❌

# Conclusion

Takeaway: **Functoriality of type formers structures type casts**

Done ✓:

- Type theory that exhibit functorial properties of type formers
- Formalised in Agda as a QIIT

WIP ⚠️ :

- Theory of signatures with subtyping
- More models of 2-CwFs

Future work ❌:

- Add inv/equivariance
- Links between 2-CwFs and other existing models (e.g comprehension categories)

# Bibliography

[1] M. Lennon-Bertrand, K. Maillard, N. Tabareau, and É. Tanter, "Gradualizing the Calculus of Inductive Constructions," *ACM Transactions on Programming Languages and Systems*, vol. 44, no. 2, Apr. 2022, doi: 10.1145/3495528.

[2] L. Pujet and N. Tabareau, "Observational Equality: Now for Good," *Proc. ACM Program. Lang.*, vol. 6, no. POPL, 2022, doi: 10.1145/3498693.

[3] T. Laurent, M. Lennon-Bertrand, and K. Maillard, "Definitional Functoriality for Dependent (Sub)Types," in *33rd European Symposium on Programming, ESOP 2024*, S. Weirich, Ed., in Lecture Notes in Computer Science, vol. 14576. Springer, 2024, pp. 302–331. doi: 10.1007/978-3-031-57262-3_13.

[4] C. McBride and F. Nordvall Forsberg, "Functorial Adapters," in *27th International Conference on Types for Proofs and Programs*, 2021.