Appalachian State University

Pierre Cagne

joint work with Patricia Johann

Appalachian State University

Partiality wrecks GADTs' functoriality

How to get Initial Algebra Semantics for GADTs?

TYPES 2023 – València

June 14th, 2023

1. First-order IAS for ADTs

2. Higher-order IAS for ADTs

3. IAS for GADTs?

1. First-order IAS for ADTs

ADTs

ADTs are families of inductive types:

ADTs

ADTs are families of inductive types:

```
data List (α : Set) : Set where
  [] : List α
  _::_ : α → List α → List α
```

ADTs

ADTs are families of inductive types:

```
data List (α : Set) : Set where
  [] : List α
  _::_ : α → List α → List α
```

```
data BinTree (α : Set) : Set where
  ∅ : BinTree α
  _⊗_⊗_ : BinTree α → α → BinTree α → BinTree α
```

ADTs

ADTs are families of inductive types:

```
data List (α : Set) : Set where
  [] : List α
  _::_ : α → List α → List α

data BinTree (α : Set) : Set where
  ∅ : BinTree α
  _⊗_⊗_ : BinTree α → α → BinTree α → BinTree α

data ℕ : Set where
  zero : ℕ
  succ : ℕ → ℕ
```

Categorical semantics of ADTs

```
data List (α : Set) : Set where
  [] : ⊤ → List α
  _::_ : α → List α → List α
```

Categorical semantics of ADTs

```
data List (α : Set) : Set where
  [] : ⊤ → List α
  _::_ : α → List α → List α
```

$\mathcal{C}$: category with finite products, finite sums, and colimits of ω-chains.

To interpret `List A`, take the initial algebra μL_A of:

$$\begin{aligned} L_A &: \mathcal{C} \to \mathcal{C} \\ & X \mapsto 1 + (A \times X) \end{aligned}$$

where A interprets `A`

Categorical semantics of ADTs

ADTs are *uniform* families of inductive types:

```
data List (α : Set) : Set where
  [] : ⊤ → List α
  _::_ : α → List α → List α
```

Categorical semantics of ADTs

ADTs are *uniform* families of inductive types:

```
data List (α : Set) : Set where
  [] : ⊤ → List α
  _∷_ : α → List α → List α
```

$$\mathcal{C} \xrightarrow{L} [\mathcal{C}, \mathcal{C}]_\omega \xrightarrow{\mu} \mathcal{C}$$

$$A \longmapsto L_A \longmapsto \mu L_A$$

2. Higher-order IAS for ADTs

Categorical semantics of ADTs

ADTs can be seen as inductive families of types:

```
data List : Set → Set where
  [] : ∀ α → ⊤ → List α
  _::_ : ∀ α → α → List α → List α
```

Categorical semantics of ADTs

ADTs can be seen as inductive families of types:

```
data List : Set → Set where
  [] : ∀ α → ⊤ → List α
  _::_ : ∀ α → α → List α → List α
```

Rework the semantics: to interpret the type constructor `List`, take the initial algebra $\mu\mathcal{L}$ of

$$\mathcal{L} : [\mathcal{C}, \mathcal{C}] \to [\mathcal{C}, \mathcal{C}]$$
$$F \mapsto (X \mapsto 1 + (X \times F(X))$$

Categorical semantics of ADTs

ADTs can be seen as inductive families of types:

```
data List : Set → Set where
  [] : ∀ α → ⊤ → List α
  _::_ : ∀ α → α → List α → List α
```

Rework the semantics: to interpret the type constructor `List`, take the initial algebra $\mu\mathcal{L}$ of

$$\begin{aligned}\mathcal{L} : [\mathcal{C}, \mathcal{C}] &\to [\mathcal{C}, \mathcal{C}] \\ F &\mapsto (X \mapsto 1 + (X \times F(X))\end{aligned}$$

All is well

$\mu\mathcal{L} \simeq \mu \circ L.$

3. IAS for GADTs?

Generalized Algebraic Data Types

GADTs are inductive families of types, with a twist:

```
data Terms : Set → Set where
  nat : ℕ → Terms ℕ
  _,_ : ∀ {α β} → Terms α → Terms β → Terms (α × β)
  π₁  : ∀ {α β} → Terms (α × β) → Terms α
  π₂  : ∀ {α β} → Terms (α × β) → Terms β
```

Generalized Algebraic Data Types

GADTs are inductive families of types, with a twist:

```
data Terms : Set → Set where
  nat  : ℕ → Terms ℕ
  _,_  : ∀ {α β} → Terms α → Terms β → Terms (α × β)
  π₁  : ∀ {α β} → Terms (α × β) → Terms α
  π₂  : ∀ {α β} → Terms (α × β) → Terms β
```

```
data W : Set → Set where
  ∃ : ∀ α → α → W ⊤
```

Generalized Algebraic Data Types

GADTs are inductive families of types, with a twist:

```
data Terms : Set → Set where
  nat : ℕ → Terms ℕ
  _,_ : ∀ {α β} → Terms α → Terms β → Terms (α × β)
  π₁ : ∀ {α β} → Terms (α × β) → Terms α
  π₂ : ∀ {α β} → Terms (α × β) → Terms β
```

```
data W : Set → Set where
  ∃ : ∀ α → α → W ⊤
```

```
data _≡_ : Set → Set → Set where
  r : ∀ α → α ≡ α
```

Naive categorical semantics of GADTs

```
data Terms : Set → Set where
  nat : ℕ → Terms ℕ
  _,_ : ∀ {α β} → Terms α → Terms β → Terms (α × β)
  π₁  : ∀ {α β} → Terms (α × β) → Terms α
  π₂  : ∀ {α β} → Terms (α × β) → Terms β
```

Naive categorical semantics of GADTs

```
data Terms : Set → Set where
  nat : ℕ → Terms ℕ
  _,_ : ∀ {α β} → Terms α → Terms β → Terms (α × β)
  π₁  : ∀ {α β} → Terms (α × β) → Terms α
  π₂  : ∀ {α β} → Terms (α × β) → Terms β
```

To interpret the type constructor `Terms`, take the initial algebra $\mu\mathcal{T}$ of:

$$\mathcal{T} : [\mathcal{C}, \mathcal{C}] \to [\mathcal{C}, \mathcal{C}]$$

$$F \mapsto \left(X \mapsto \quad + \quad + \quad + \quad \right)$$

Naive categorical semantics of GADTs

```
data Terms : Set → Set where
  nat : ℕ → Terms ℕ
  _,_ : ∀ {α β} → Terms α → Terms β → Terms (α × β)
  π₁ : ∀ {α β} → Terms (α × β) → Terms α
  π₂ : ∀ {α β} → Terms (α × β) → Terms β
```

To interpret the type constructor `Terms`, take the initial algebra $\mu\mathcal{T}$ of:

$$\mathcal{T} : [\mathcal{C}, \mathcal{C}] \to [\mathcal{C}, \mathcal{C}]$$

$$F \mapsto \begin{pmatrix} X \mapsto \quad N \text{ if } X = N,\ \emptyset \text{ otherwise} \\ + \\ \\ + \\ \\ + \\ \end{pmatrix}$$

Naive categorical semantics of GADTs

```
data Terms : Set → Set where
  nat : ℕ → Terms ℕ
  _,_ : ∀ {α β} → Terms α → Terms β → Terms (α × β)
  π₁  : ∀ {α β} → Terms (α × β) → Terms α
  π₂  : ∀ {α β} → Terms (α × β) → Terms β
```

To interpret the type constructor `Terms`, take the initial algebra $\mu\mathcal{T}$ of:

$$\mathcal{T} : [\mathcal{C}, \mathcal{C}] \to [\mathcal{C}, \mathcal{C}]$$

$$F \mapsto \left(X \mapsto \begin{array}{l} N \text{ if } X = N, \varnothing \text{ otherwise} \\ + \sum_{\substack{X_1, X_2 \\ X_1 \times X_2 = X}} F(X_1) \times F(X_2) \\ + \\ + \end{array} \right)$$

Naive categorical semantics of GADTs

```
data Terms : Set → Set where
  nat : ℕ → Terms ℕ
  _,_ : ∀ {α β} → Terms α → Terms β → Terms (α × β)
  π₁  : ∀ {α β} → Terms (α × β) → Terms α
  π₂  : ∀ {α β} → Terms (α × β) → Terms β
```

To interpret the type constructor `Terms`, take the initial algebra $\mu\mathcal{T}$ of:

$$\mathcal{T} : [\mathcal{C}, \mathcal{C}] \to [\mathcal{C}, \mathcal{C}]$$

$$F \mapsto \begin{pmatrix} X \mapsto & N \text{ if } X = N, \varnothing \text{ otherwise} \\ & + \sum_{\substack{X_1, X_2 \\ X_1 \times X_2 = X}} F(X_1) \times F(X_2) \\ & + \sum_{X_2} F(X \times X_2) \\ & + \end{pmatrix}$$

Naive categorical semantics of GADTs

```
data Terms : Set → Set where
  nat : ℕ → Terms ℕ
  _,_ : ∀ {α β} → Terms α → Terms β → Terms (α × β)
  π₁  : ∀ {α β} → Terms (α × β) → Terms α
  π₂  : ∀ {α β} → Terms (α × β) → Terms β
```

To interpret the type constructor `Terms`, take the initial algebra $\mu\mathcal{T}$ of:

$$\mathcal{T} : [\mathcal{C}, \mathcal{C}] \to [\mathcal{C}, \mathcal{C}]$$

$$F \mapsto \left(X \mapsto \begin{array}{l} N \text{ if } X = N,\ \varnothing \text{ otherwise} \\ + \displaystyle\sum_{\substack{X_1, X_2 \\ X_1 \times X_2 = X}} F(X_1) \times F(X_2) \\ + \displaystyle\sum_{X_2} F(X \times X_2) \\ + \displaystyle\sum_{X_1} F(X_1 \times X) \end{array} \right)$$

Naive categorical semantics of GADTs

```
data Terms : Set → Set where
  nat : ℕ → Terms ℕ
  _,_ : ∀ {α β} → Terms α → Terms β → Terms (α × β)
  π₁  : ∀ {α β} → Terms (α × β) → Terms α
  π₂  : ∀ {α β} → Terms (α × β) → Terms β
```

To interpret the type constructor `Terms`, take the initial algebra $\mu\mathcal{T}$ of:

$$\mathcal{T} : [\mathcal{C}, \mathcal{C}] \to [\mathcal{C}, \mathcal{C}]$$

$$F \mapsto \left(X \mapsto \begin{array}{ll} N \text{ if } X = N,\ \emptyset \text{ otherwise} & \\ + \displaystyle\sum_{\substack{X_1, X_2 \\ X_1 \times X_2 = X}} F(X_1) \times F(X_2) & \\ + \displaystyle\sum_{X_2} F(X \times X_2) & ?? \\ + \displaystyle\sum_{X_1} F(X_1 \times X) & ?? \end{array} \right)$$

Naive categorical semantics of GADTs

```
data Terms : Set → Set where
  nat : ℕ → Terms ℕ
  _,_ : ∀ {α β} → Terms α → Terms β → Terms (α × β)
  π₁ : ∀ {α β} → Terms (α × β) → Terms α
  π₂ : ∀ {α β} → Terms (α × β) → Terms β
```

Actually, $\mu\mathcal{T} : \mathcal{C} \to \mathcal{C}$ being a functor is already an issue.

Naive categorical semantics of GADTs

```
data Terms : Set → Set where
  nat : ℕ → Terms ℕ
  _,_ : ∀ {α β} → Terms α → Terms β → Terms (α × β)
  π₁ : ∀ {α β} → Terms (α × β) → Terms α
  π₂ : ∀ {α β} → Terms (α × β) → Terms β
```

Actually, $\mu\mathcal{T} : \mathcal{C} \to \mathcal{C}$ being a functor is already an issue.

Consider the parity function `p: ℕ → 𝔹`, interpreted by $p : N \to B$. Because of $\mu\mathcal{T}(p) : \mu\mathcal{T}(N) \to \mu\mathcal{T}(B)$, the interpretation of `Terms 𝔹` contains weird elements...

Naive categorical semantics of GADTs

Even more striking with

```
data W : Set → Set where
  ∃ : ∀ {α} → α → W ⊤
```

Naive categorical semantics of GADTs

Even more striking with

```
data W : Set → Set where
  ∃ : ∀ {α} → α → W ⊤
```

If `W` is interpreted as the initial algebra $\mu\mathcal{W} : \mathcal{C} \to \mathcal{C}$ of a certain $\mathcal{W}$:

Naive categorical semantics of GADTs

Even more striking with

```
data W : Set → Set where
  ∃ : ∀ {α} → α → W ⊤
```

If `W` is interpreted as the initial algebra $\mu\mathcal{W} : \mathcal{C} \to \mathcal{C}$ of a certain $\mathcal{W}$:

- for each $X \in \mathcal{C}$,

$$\exists_X : X \to \mu\mathcal{W}(1),$$

Naive categorical semantics of GADTs

Even more striking with

```
data W : Set → Set where
  ∃ : ∀ {α} → α → W ⊤
```

If `W` is interpreted as the initial algebra $\mu\mathcal{W} : \mathcal{C} \to \mathcal{C}$ of a certain $\mathcal{W}$:

- for each $X \in \mathcal{C}$,

$$\exists_X : X \to \mu\mathcal{W}(1),$$

- for each $x : 1 \to X$,

$$1 \xrightarrow{\exists_1} \mu\mathcal{W}(1) \xrightarrow{\mu\mathcal{W}(x)} \mu\mathcal{W}(X)$$

Naive categorical semantics of GADTs

Even more striking with

```
data W : Set → Set where
  ∃ : ∀ {α} → α → W ⊤
```

If `W` is interpreted as the initial algebra $\mu\mathcal{W} : \mathcal{C} \to \mathcal{C}$ of a certain $\mathcal{W}$:

- for each $X \in \mathcal{C}$,

$$\exists_X : X \to \mu\mathcal{W}(1),$$

- for each $x : 1 \to X$,

$$1 \xrightarrow{\exists_1} \mu\mathcal{W}(1) \xrightarrow{\mu\mathcal{W}(x)} \mu\mathcal{W}(X)$$

When $\mathcal{C} = \mathbf{Set}$, $\mu\mathcal{W}(X) \neq \emptyset$ whenever $X \neq \emptyset$...

Naive categorical semantics of GADTs

$$1 \xrightarrow{\exists_1} \mu\mathcal{W}(1) \xrightarrow{\mu\mathcal{W}(x)} \mu\mathcal{W}(X)$$

Naive categorical semantics of GADTs

$$1 \xrightarrow{\exists_1} \mu\mathcal{W}(1) \xrightarrow{\mu\mathcal{W}(x)} \mu\mathcal{W}(X)$$

Issue

$\mu\mathcal{W}(x)$ has to make a new element from $\exists_1$.

Naive categorical semantics of GADTs

$$1 \xrightarrow{\exists_1} \mu\mathcal{W}(1) \xrightarrow{\mu\mathcal{W}(x)} \mu\mathcal{W}(X)$$

Issue

$\mu\mathcal{W}(x)$ has to make a new element from $\exists_1$.

Potential solution

Allowing $\mu\mathcal{W}(x)$ to be only partially defined.

Less naive categorical semantics of GADTs

Definition

A structure of partiality on $\mathcal{C}$ is a wide subcategory $\mathcal{D}$ such that

$$\forall f, g \in \mathcal{C}, gf \in \mathcal{D} \implies f \in \mathcal{D}$$

Less naive categorical semantics of GADTs

Definition

A structure of partiality on $\mathcal{C}$ is a wide subcategory $\mathcal{D}$ such that

$$\forall f, g \in \mathcal{C}, gf \in \mathcal{D} \implies f \in \mathcal{D}$$

Example

- $\mathcal{D} = \mathcal{C}$ (trivial)

Less naive categorical semantics of GADTs

Definition

A structure of partiality on $\mathcal{C}$ is a wide subcategory $\mathcal{D}$ such that

$$\forall f, g \in \mathcal{C}, gf \in \mathcal{D} \implies f \in \mathcal{D}$$

Example

- $\mathcal{D} = \mathcal{C}$ (trivial)
- $\mathcal{C} = \mathsf{PSet}$ and $\mathcal{D} = \mathsf{Set}$

Less naive categorical semantics of GADTs

Definition

A structure of partiality on $\mathcal{C}$ is a wide subcategory $\mathcal{D}$ such that

$$\forall f, g \in \mathcal{C}, gf \in \mathcal{D} \implies f \in \mathcal{D}$$

Example

- $\mathcal{D} = \mathcal{C}$ (trivial)
- $\mathcal{C} = \mathsf{PSet}$ and $\mathcal{D} = \mathsf{Set}$
- $\mathcal{C} = \mathsf{CPO}$ and $\mathcal{D} = \mathsf{CPO}_{\mathrm{tot}}$

Less naive categorical semantics of GADTs

Definition

A structure of partiality on $\mathcal{C}$ is a wide subcategory $\mathcal{D}$ such that

$$\forall f, g \in \mathcal{C}, gf \in \mathcal{D} \implies f \in \mathcal{D}$$

Example

- $\mathcal{D} = \mathcal{C}$ (trivial)
- $\mathcal{C} = \mathsf{PSet}$ and $\mathcal{D} = \mathsf{Set}$
- $\mathcal{C} = \mathsf{CPO}$ and $\mathcal{D} = \mathsf{CPO}_{\mathrm{tot}}$

Idea

Interpret the total functions of the language in $\mathcal{D}$ and “spill” in $\mathcal{C}$ for partial functions.

Less naive categorical semantics of GADTs

But...

Less naive categorical semantics of GADTs

But...

```
data _≡_ : Set → Set → Set where
  r : ∀ α → α ≡ α
```

Less naive categorical semantics of GADTs

But...

```
data _≡_ : Set → Set → Set where
  r : ∀ α → α ≡ α
```

- _≡_ interpreted as: an object $(X \equiv Y)$ for every $X, Y \in \mathcal{C}$

Less naive categorical semantics of GADTs

But...

```
data _≡_ : Set → Set → Set where
  r : ∀ α → α ≡ α
```

- `_≡_` interpreted as: an object $(X \equiv Y)$ for every $X, Y \in \mathcal{C}$
- r interpreted as: a morphism $r_X : 1 \to (X \equiv X)$ in $\mathcal{D}$ for every $X \in \mathcal{C}$

Less naive categorical semantics of GADTs

But...

```
data _≡_ : Set → Set → Set where
  r : ∀ α → α ≡ α
```

- _≡_ interpreted as: an object $(X \equiv Y)$ for every $X, Y \in \mathcal{C}$
- r interpreted as: a morphism $r_X : 1 \to (X \equiv X)$ in $\mathcal{D}$ for every $X \in \mathcal{C}$

If $\_ \equiv \_$ extends to a functor $\mathcal{C}^2 \to \mathcal{C}$: for any $x : 1 \to X$,

Less naive categorical semantics of GADTs

But...

```
data _≡_ : Set → Set → Set where
  r : ∀ α → α ≡ α
```

- _≡_ interpreted as: an object $(X \equiv Y)$ for every $X, Y \in \mathcal{C}$
- r interpreted as: a morphism $r_X : 1 \to (X \equiv X)$ in $\mathcal{D}$ for every $X \in \mathcal{C}$

If $\_ \equiv \_$ extends to a functor $\mathcal{C}^2 \to \mathcal{C}$: for any $x : 1 \to X$,

$$p_x : \quad 1 \xrightarrow{r_1} 1 \equiv 1 \xrightarrow{(x \equiv \mathrm{id}_1)} (X \equiv 1)$$

Less naive categorical semantics of GADTs

```
trp : ∀ α β → α ≡ β → α → β
trp α α r x = x

trp⁻¹ : ∀ α β → α ≡ β → β → α
trp β β r y = y
```

Less naive categorical semantics of GADTs

```
trp : ∀ α β → α ≡ β → α → β
trp α α r x = x

trp⁻¹ : ∀ α β → α ≡ β → β → α
trp β β r y = y
```

- `trp` interpreted as: a morphism $t_{X,Y} : (X \equiv Y) \times X \to Y$ in $\mathcal{D}$ for every $X, Y \in \mathcal{C}$

Less naive categorical semantics of GADTs

```
trp : ∀ α β → α ≡ β → α → β
trp α α r x = x

trp⁻¹ : ∀ α β → α ≡ β → β → α
trp β β r y = y
```

- `trp` interpreted as: a morphism $t_{X,Y} : (X \equiv Y) \times X \to Y$ in $\mathcal{D}$ for every $X, Y \in \mathcal{C}$

Less naive categorical semantics of GADTs

```
trp : ∀ α β → α ≡ β → α → β
trp α α r x = x

trp⁻¹ : ∀ α β → α ≡ β → β → α
trp β β r y = y
```

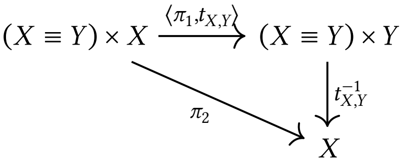
- `trp` interpreted as: a morphism $t_{X,Y} : (X \equiv Y) \times X \to Y$ in $\mathcal{D}$ for every $X, Y \in \mathcal{C}$
- `trp⁻¹` interpreted as: a morphism $t^{-1}_{X,Y} : (X \equiv Y) \times Y \to X$ in in $\mathcal{D}$ for every $X, Y \in \mathcal{C}$

Less naive categorical semantics of GADTs

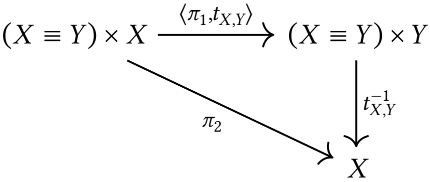
```
trp : ∀ α β → α ≡ β → α → β
trp α α r x = x

trp⁻¹ : ∀ α β → α ≡ β → β → α
trp β β r y = y
```

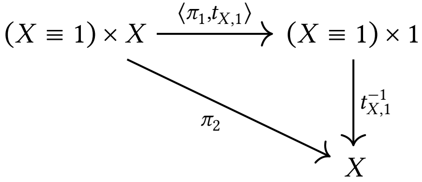
- `trp` interpreted as: a morphism $t_{X,Y} : (X \equiv Y) \times X \to Y$ in $\mathcal{D}$ for every $X, Y \in \mathcal{C}$
- `trp⁻¹` interpreted as: a morphism $t^{-1}_{X,Y} : (X \equiv Y) \times Y \to X$ in in $\mathcal{D}$ for every $X, Y \in \mathcal{C}$
- `(λ p x → trp⁻¹ p (trp p x))` reduces to `(λ p x → x)`:



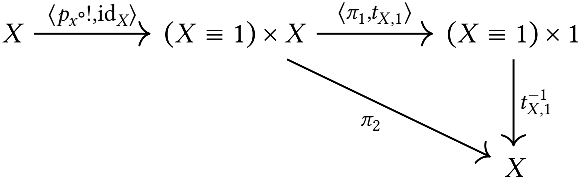
Less naive categorical semantics of GADTs



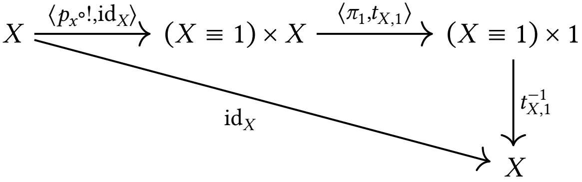
Less naive categorical semantics of GADTs



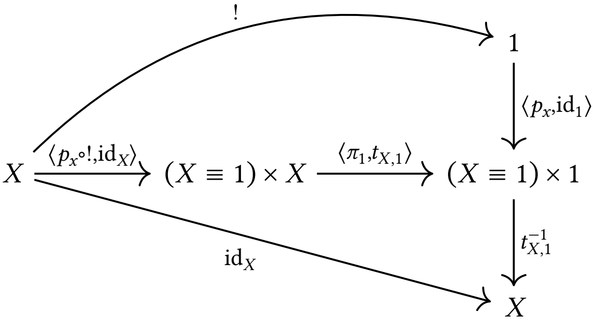
Less naive categorical semantics of GADTs



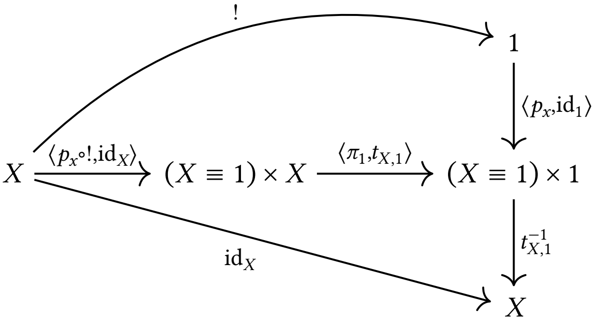
Less naive categorical semantics of GADTs



Less naive categorical semantics of GADTs

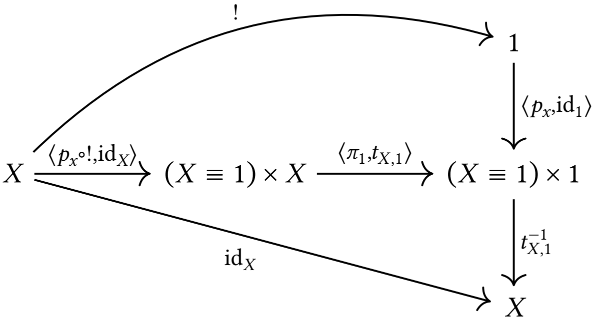


Less naive categorical semantics of GADTs



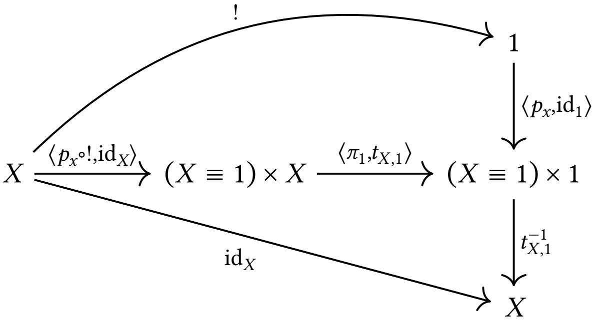
$\Longrightarrow$ X is a retract of 1

Less naive categorical semantics of GADTs



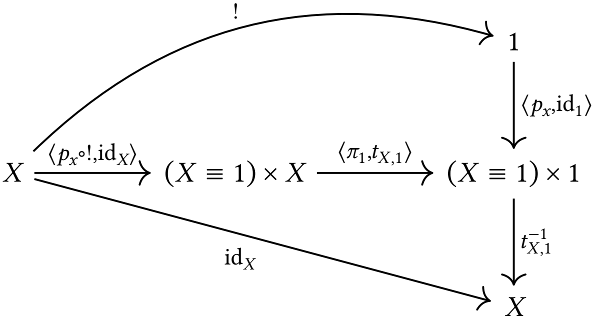
$\Longrightarrow \; X \simeq 1$ whenever $t^{-1}_{X,1}\langle p_x, \mathrm{id}_1\rangle$ in $\mathcal{D}$

Less naive categorical semantics of GADTs



$\implies X \simeq 1$ whenever p_x in $\mathcal{D}$

Less naive categorical semantics of GADTs



$\Longrightarrow$ $X \simeq 1$ whenever x in $\mathcal{D}$

Less naive categorical semantics of GADTs

Theorem

If GADTs' interpretations extend to functors, the interpretation of any non-empty closed type is trivial.

Thank you.