Appalachian State University

Pierre Cagne
joint work with Patricia Johann
Appalachian State University

GADTs aren't (even lax) functors

SMP 2023 – Boston
June 25th, 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
```

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

$$\begin{aligned} L_A \;:\; & \mathsf{Set} \to \mathsf{Set} \\ & X \mapsto 1 + (A \times X) \end{aligned}$$

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
```

$$\mathsf{Set} \xrightarrow{L} [\mathsf{Set}, \mathsf{Set}]_\omega \xrightarrow{\mu} \mathsf{Set}$$

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

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

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

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

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

All is well

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

Categorical semantics of ADTs

$\mu\mathcal{L}$ can be computed as a colimit of an ω-chain:

$$0 \to \mathcal{L}(0) \to \cdots \to \mathcal{L}^n(0) \to \cdots$$

Categorical semantics of ADTs

$\mu\mathcal{L}$ can be computed as a colimit of an ω-chain:

$$0 \to \mathcal{L}(0) \to \cdots \to \mathcal{L}^n(0) \to \cdots$$

Consequence

When A is the set of closed terms of a given type `A`,

$$\mu\mathcal{L}(A) \simeq \{\text{closed terms of type } \texttt{List A}\}$$

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:

$$\mathcal{T} \;:\; [\mathsf{Set}, \mathsf{Set}] \to [\mathsf{Set}, \mathsf{Set}]$$

$$F \mapsto \begin{pmatrix} X \mapsto \\ \quad + \\ \\ \quad + \\ \quad + \\ \end{pmatrix}$$

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:

$$\mathcal{T} : [\mathrm{Set}, \mathrm{Set}] \to [\mathrm{Set}, \mathrm{Set}]$$

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

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:

$$\mathcal{T} : [\mathrm{Set}, \mathrm{Set}] \to [\mathrm{Set}, \mathrm{Set}]$$

$$F \mapsto \left(X \mapsto \mathbb{N} \text{ if } X = \mathbb{N},\ \emptyset \text{ otherwise} + \sum_{\substack{X_1, X_2 \\ X_1 \times X_2 = X}} F(X_1) \times F(X_2) + \quad + \quad \right)$$

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:

$$\mathcal{T} : [\mathsf{Set}, \mathsf{Set}] \to [\mathsf{Set}, \mathsf{Set}]$$

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

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:

$$\mathcal{T} : [\mathsf{Set}, \mathsf{Set}] \to [\mathsf{Set}, \mathsf{Set}]$$

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

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:

$$\mathcal{T} : [\mathsf{Set}, \mathsf{Set}] \to [\mathsf{Set}, \mathsf{Set}]$$

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

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, $\mu\mathfrak{T} : \mathsf{Set} \to \mathsf{Set}$ 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, $\mu\mathfrak{T} : \mathsf{Set} \to \mathsf{Set}$ being a functor is already an issue.

Consider the parity function `p: ℕ → 𝔹`, interpreted by $p : \mathbb{N} \to \mathbb{B}$. Because of $\mu\mathfrak{T}(p) : \mu\mathfrak{T}(\mathbb{N}) \to \mu\mathfrak{T}(\mathbb{B})$, 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 $\mu\mathcal{W} : \mathsf{Set} \to \mathsf{Set}$ 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 $\mu\mathcal{W} : \mathsf{Set} \to \mathsf{Set}$ of a certain $\mathcal{W}$:

- for each $X \in \mathsf{Set}$,

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

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 $\mu\mathcal{W} : \mathbf{Set} \to \mathbf{Set}$ of a certain $\mathcal{W}$:

- for each $X \in \mathbf{Set}$,

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

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

$$\mu\mathcal{W}(X) \ni \mu\mathcal{W}(x)(\exists_1(*))$$

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 $\mu\mathcal{W} : \mathbf{Set} \to \mathbf{Set}$ of a certain $\mathcal{W}$:

- for each $X \in \mathbf{Set}$,

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

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

$$\mu\mathcal{W}(X) \ni \mu\mathcal{W}(x)(\exists_1(*))$$

That is, $\mu\mathcal{W}(X) \neq \emptyset$ 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 in $\mu\mathcal{W}(X)$ from $\exists_1(*) \in \mu\mathcal{W}(1)$.

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 in $\mu\mathcal{W}(X)$ from $\exists_1(*) \in \mu\mathcal{W}(1)$.

Potential solution

Allowing $\mu\mathcal{W}(x)$ to be a partial function.

Less naive categorical semantics of GADTs

PSet: category of sets and partial function between them. So $\mathbf{Set} \hookrightarrow \mathbf{PSet}$.

Less naive categorical semantics of GADTs

PSet: category of sets and partial function between them. So $\mathbf{Set} \hookrightarrow \mathbf{PSet}$.

Fact

For any compasable functions f, g in **PSet**, if $g\,f$ is total, then so is f.

Less naive categorical semantics of GADTs

PSet: category of sets and partial function between them. So $\mathbf{Set} \hookrightarrow \mathbf{PSet}$.

Fact

For any compasable functions f, g in **PSet**, if gf is total, then so is f.

Idea

Interpret the total functions of the language in **Set** and “spill” in **PSet** 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: a set $(X \equiv Y)$ for every sets X, Y

Less naive categorical semantics of GADTs

But...

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

- `_≡_` interpreted as: a set $(X \equiv Y)$ for every sets X, Y
- r interpreted as: a total function $r_X : 1 \to (X \equiv X)$ for every set X

Less naive categorical semantics of GADTs

But...

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

- _≡_ interpreted as: a set $(X \equiv Y)$ for every sets X, Y
- r interpreted as: a total function $r_X : 1 \to (X \equiv X)$ for every set X

If $- \equiv -$ extends to a functor $\mathbf{PSet}^2 \to \mathbf{PSet}$: for any $x : 1 \to X$,

Less naive categorical semantics of GADTs

But...

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

- _≡_ interpreted as: a set $(X \equiv Y)$ for every sets X, Y
- r interpreted as: a total function $r_X : 1 \to (X \equiv X)$ for every set X

If $- \equiv -$ extends to a functor $\mathbf{PSet}^2 \to \mathbf{PSet}$: 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 total function $t_{X,Y} : (X \equiv Y) \times X \to Y$ for all sets X, 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 total function $t_{X,Y} : (X \equiv Y) \times X \to Y$ for all sets X, Y
- `trp⁻¹` interpreted as: a total function $t^{-1}_{X,Y} : (X \equiv Y) \times Y \to X$ for all sets X, 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
```

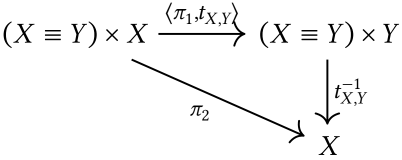
- `trp` interpreted as: a total function $t_{X,Y} : (X \equiv Y) \times X \to Y$ for all sets X, Y
- `trp⁻¹` interpreted as: a total function $t^{-1}_{X,Y} : (X \equiv Y) \times Y \to X$ for all sets X, Y

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
```

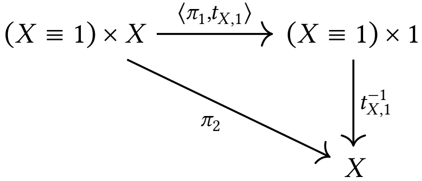
- `trp` interpreted as: a total function $t_{X,Y} : (X \equiv Y) \times X \to Y$ for all sets X, Y
- `trp⁻¹` interpreted as: a total function $t^{-1}_{X,Y} : (X \equiv Y) \times Y \to X$ for all sets X, Y
- `(λ p x → trp⁻¹ p (trp p x))` reduces to `(λ p x → x)`:



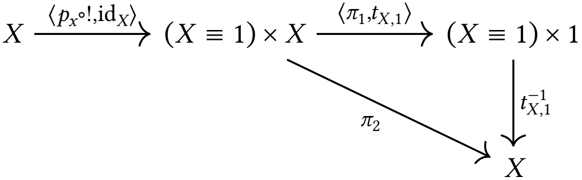
Less naive categorical semantics of GADTs

$$
\begin{array}{ccc}
(X \equiv Y) \times X & \xrightarrow{\langle \pi_1, t_{X,Y} \rangle} & (X \equiv Y) \times Y \\
 & \searrow^{\pi_2} & \downarrow t^{-1}_{X,Y} \\
 & & X
\end{array}
$$

Less naive categorical semantics of GADTs

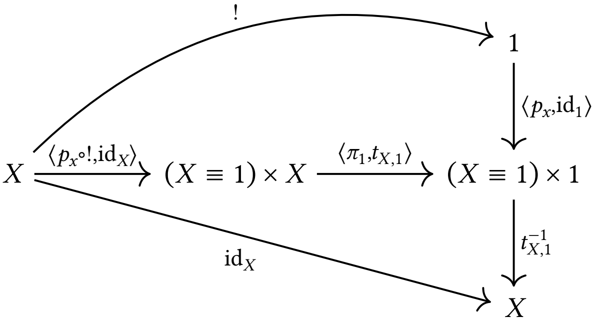


Less naive categorical semantics of GADTs

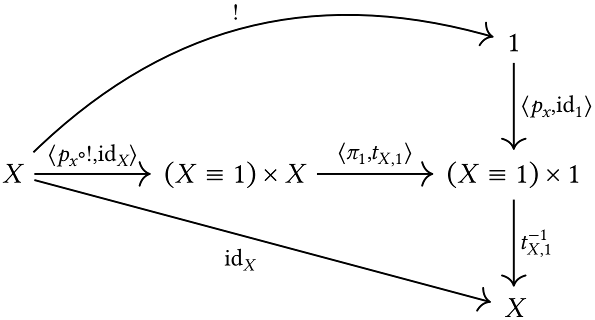


$$
\begin{array}{ccccc}
X & \xrightarrow{\langle p_x \circ !, \mathrm{id}_X \rangle} & (X \equiv 1) \times X & \xrightarrow{\langle \pi_1, t_{X,1} \rangle} & (X \equiv 1) \times 1 \\
 & & & \searrow^{\mathrm{id}_X} & \downarrow t^{-1}_{X,1} \\
 & & & & X
\end{array}
$$

Less naive categorical semantics of GADTs

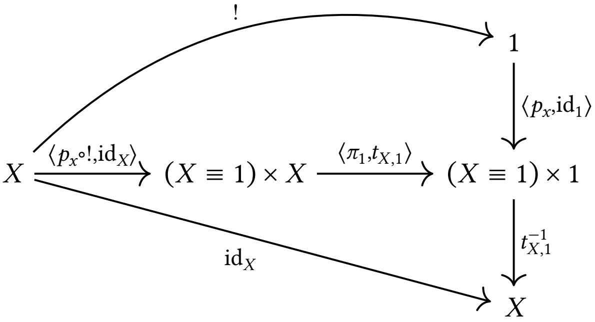


Less naive categorical semantics of GADTs



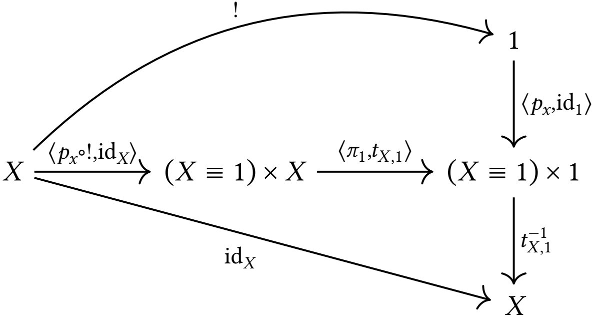
$\Longrightarrow$ X is a retract of 1

Less naive categorical semantics of GADTs



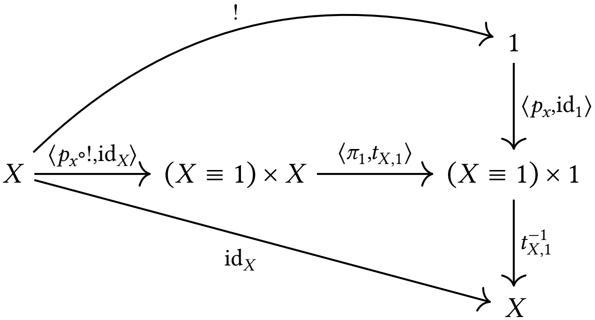
$\implies X \simeq 1$ whenever $t_{X,1}^{-1} \langle p_x, \mathrm{id}_1 \rangle$ is total

Less naive categorical semantics of GADTs



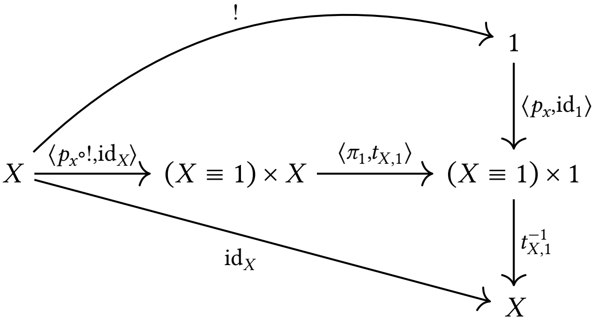
$\Longrightarrow$ $X \simeq 1$ whenever p_x is total

Less naive categorical semantics of GADTs



$\Longrightarrow$ $X \simeq 1$ whenever x is total

Less naive categorical semantics of GADTs



$\implies$ $X \simeq 1$ whenever $X \neq \emptyset$

Less naive categorical semantics of GADTs

Theorem

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

Less naive categorical semantics of GADTs

Theorem

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

Issue

Functors send sections to sections, but GADTs send sections to partial injections.

In **PSet**: $f \leq g$ if and only if $\mathrm{Dom}\, f \subseteq \mathrm{Dom}\, g$ is more defined than f and $g \restriction_{\mathrm{Dom}\, f} = f$.

Lax-functorial semantics of GADTs

In **PSet**: $f \leq g$ if and only if $\mathrm{Dom}\ f \subseteq \mathrm{Dom}\ g$ is more defined than f and $g \restriction_{\mathrm{Dom}\ f} = f$.

Definition

A normal lax functor $G : \mathsf{PSet} \to \mathsf{PSet}$ is just like a functor except:

- G respects $\leq$,
- $G(gf) \leq G(g)G(f)$ instead of $G(gf) = G(g)G(f)$.

Lax-functorial semantics of GADTs

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

But…

Lax-functorial semantics of GADTs

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

But...

Consider $f, g : \mathbb{N} \times \mathbb{N} \to \mathbb{N} \times \mathbb{N}$ with:

$$f(0, y) = (0, y), \qquad f(x > 0, y) \text{ undefined}$$
$$g(x, y) = (x, x + y)$$

If $T : \mathsf{PSet} \to \mathsf{PSet}$ normal lax interprets `Terms`, then $T(f) \leq T(g)$.

Lax-functorial semantics of GADTs

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

- $n : \mathbb{N} \to T(\mathbb{N})$ interprets `nat`
- $\langle -, - \rangle_{X,Y} : T(X) \times T(Y) \to T(X \times Y)$ interprets `_,_` at X, Y

Lax-functorial semantics of GADTs

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

- $n : \mathbb{N} \to T(\mathbb{N})$ interprets `nat`
- $\langle -, - \rangle_{X,Y} : T(X) \times T(Y) \to T(X \times Y)$ interprets `_,_` at X, Y

- $T(f)(\langle n(0), n(y) \rangle) = \langle n(0), n(y) \rangle$

Lax-functorial semantics of GADTs

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

- $n : \mathbb{N} \to T(\mathbb{N})$ interprets `nat`
- $\langle -, - \rangle_{X,Y} : T(X) \times T(Y) \to T(X \times Y)$ interprets `_,_` at X, Y

- $T(f)(\langle n(0), n(y) \rangle) = \langle n(0), n(y) \rangle$
- $T(f) \leq T(g)$ so $T(g)$ also defined on $\langle n(0), n(y) \rangle$

Lax-functorial semantics of GADTs

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

- $n : \mathbb{N} \to T(\mathbb{N})$ interprets `nat`
- $\langle -, - \rangle_{X,Y} : T(X) \times T(Y) \to T(X \times Y)$ interprets `_,_` at X, Y

- $T(f)(\langle n(0), n(y) \rangle) = \langle n(0), n(y) \rangle$
- $T(f) \leq T(g)$ so $T(g)$ also defined on $\langle n(0), n(y) \rangle$
- $T(g)(\langle n(0), n(y) \rangle) = T(f)(\langle n(0), n(y) \rangle) = \langle n(0), n(y) \rangle$...

Thank you.