Type theory as a topic of mathematical study

An introduction to Generalised Algebraic Theories

Formal proof and synthetic mathematics workshop – June 26th 2026

Meven Lennon-Bertrand

Inria – IRIF, Université Paris Cité

A bit of context

We've been doing quite a bit of type theory...
Or, to be precise, worked *in* type theories.

A bit of context

We've been doing quite a bit of type theory...
Or, to be precise, worked *in* type theories.

But:

- What is a type theory?
- What is a model of such a type theory?
- What does it mean "to work in" such a type theory?

A bit of context

We've been doing quite a bit of type theory...
Or, to be precise, worked *in* type theories.

But:

- What is a type theory?
- What is a model of such a type theory?
- What does it mean "to work in" such a type theory?

These questions can be given mathematical answers

Warmup: Universal algebra

Who here has been exposed to universal algebra?

Signature

Ambient logic = set-theoretic notation (but could be a type theory!): $\in$, $\mathbf{Set}$, $\prod$, $\sum$...

Signature

Ambient logic = set-theoretic notation (but could be a type theory!): $\in$, $\mathbf{Set}$, $\prod$, $\sum$...

Signature

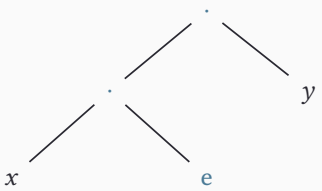
A signature Σ consists of:

- a set of operations (generators) op_Σ, specified by an arity $\mathrm{ar}_\Sigma \in \mathrm{op}_\Sigma \to \mathbb{N}$

Signature of monoids :

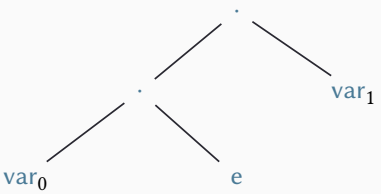
- $\mathrm{op} := \{e, \cdot\}, \quad \mathrm{ar}(e) := 0, \quad \mathrm{ar}(\cdot) := 2$

Syntax trees



- They exist!
- Nice representation in (functional) programming languages (algebraic datatypes)

Syntax trees



- They exist!
- Nice representation in (functional) programming languages (algebraic datatypes)
- Variables complicate things, but there are ways

Signature

Ambient logic = set-theoretic notation (but could be a type theory!): $\in$, **Set**, $\prod$, $\sum$...

Signature

A signature Σ consists of:

- a set of operations (generators) op_Σ, specified by an arity $\mathrm{ar}_\Sigma \in \mathrm{op}_\Sigma \to \mathbb{N}$
  $\longrightarrow$ generates a set of terms $\mathsf{Tm} \in \mathbb{N} \to \mathbf{Set}$

Signature of monoids :

- $\mathrm{op} := \{\mathrm{e}, \cdot\}, \quad \mathrm{ar}(\mathrm{e}) := 0, \quad \mathrm{ar}(\cdot) := 2$
- $\mathrm{e} \in \mathsf{Tm}(0), \quad (x \cdot \mathrm{e}) \cdot y \in \mathsf{Tm}(2)$

Signature

Ambient logic = set-theoretic notation (but could be a type theory!): $\in$, **Set**, $\prod$, $\sum$...

Signature

A signature Σ consists of:

- a set of operations (generators) op_Σ, specified by an arity $\mathrm{ar}_\Sigma \in \mathrm{op}_\Sigma \to \mathbb{N}$
  $\longrightarrow$ generates a set of terms $\mathsf{Tm} \in \mathbb{N} \to \mathbf{Set}$
- for each n, a set of equations (relations) $\mathrm{eqs}_\Sigma(n) \subseteq \mathsf{Tm}(n) \times \mathsf{Tm}(n)$

Signature of monoids :

- $\mathrm{op} := \{\mathsf{e}, \cdot\}, \quad \mathrm{ar}(\mathsf{e}) := 0, \quad \mathrm{ar}(\cdot) := 2$
- $\mathsf{e} \in \mathsf{Tm}(0), \quad (x \cdot \mathsf{e}) \cdot y \in \mathsf{Tm}(2)$
- $\mathrm{eqs}(1) := \{(x \cdot \mathsf{e} = x), (\mathsf{e} \cdot x = x)\}, \quad \mathrm{eqs}(3) := \{x \cdot (y \cdot z) = (x \cdot y) \cdot z\}$

Models

Model of a signature (sort, op, ar)

- a set X

A monoid = a model of the signature of monoids:

- a set X

Models

Model of a signature (sort, op, ar)

- a set X
- for every operation $o \in \mathrm{op}$ with a function $f_o \in X^{\mathrm{ar}(o)} \to X$

A monoid = a model of the signature of monoids:

- a set X
- a constant $e \in X$ and a function $f \in X \times X \to X$

Models

Model of a signature (sort, op, ar)

- a set X
- for every operation $\mathrm{o} \in \mathrm{op}$ with a function $f_{\mathrm{o}} \in X^{\mathrm{ar}(\mathrm{o})} \to X$
  $\longrightarrow$ lifted to $[\![\_]\!] \in \mathsf{Tm}(n) \to X^n \to X$

A monoid = a model of the signature of monoids:

- a set X
- a constant $e \in X$ and a function $f \in X \times X \to X$
- $[\![\mathrm{e}]\!] = e, \quad [\![(x \cdot \mathrm{e}) \cdot y]\!](u, v) = f(f(u, e), v)$

Models

Model of a signature (sort, op, ar)

- a set X
- for every operation $o \in \mathrm{op}$ with a function $f_o \in X^{\mathrm{ar}(o)} \to X$
  $\rightsquigarrow$ lifted to $[\![\_]\!] \in \mathrm{Tm}(n) \to X^n \to X$
- such that for every $(l, r) \in \mathrm{eqs}(n)$ and $v \in X^n$, we have $[\![l]\!](v) = [\![r]\!](v)$

A monoid = a model of the signature of monoids:

- a set X
- a constant $e \in X$ and a function $f \in X \times X \to X$
- $[\![e]\!] = e, \quad [\![(x \cdot e) \cdot y]\!](u, v) = f(f(u, e), v)$
- such that for all u, v and w we have $f(e, v) = v$, $f(v, e) = v$ and $f(u, f(v, w)) = f(f(u, v), w)$

Model of a signature (sort, op, ar)

- a set X
- for every operation $\mathrm{o} \in \mathrm{op}$ with a function $f_{\mathrm{o}} \in X^{\mathrm{ar}(\mathrm{o})} \to X$
  $\rightsquigarrow$ lifted to $[\![\_]\!] \in \mathrm{Tm}(n) \to X^n \to X$
- such that for every $(l, r) \in \mathrm{eqs}(n)$ and $v \in X^n$, we have $[\![l]\!](v) = [\![r]\!](v)$

A monoid = a model of the signature of monoids:

- a set X
- a constant $e \in X$ and a function $f \in X \times X \to X$
- $[\![\mathrm{e}]\!] = e, \quad [\![(x \cdot \mathrm{e}) \cdot y]\!](u, v) = f(f(u, e), v)$
- such that for all u, v and w we have $f(e, v) = v$, $f(v, e) = v$ and $f(u, f(v, w)) = f(f(u, v), w)$

+ **Morphism of model**

Free/initial model

Theorem (originally, Birkhoff?)

Every signature has a **free model**, which is initial amongst all models.

Free/initial model

Theorem (originally, Birkhoff?)

Every signature has a **free model**, which is initial amongst all models.

Free model for the signature of monoids	=	free monoid = $\{e\}$
Free model for the signature of monoids + 1 constant per $a \in A$	=	free monoid on A

Generalised algebraic theory

Multi-sorted universal algebra

Monoids, rings, propositional logic… fit in universal algebra
Modules, vector spaces… need multiple sorts

Monoids, rings, propositional logic... fit in universal algebra
Modules, vector spaces... need multiple sorts

Multi-sorted signature

A signature Σ consists of:

- a **set of sorts** $\mathrm{sort}_\Sigma \in \mathbf{Set}$
- a set of operations op_Σ, specified by an arity $\mathrm{ar}_\Sigma \in \mathrm{op}_\Sigma \to {\mathrm{sort}_\Sigma}^\star \times \mathrm{sort}_\Sigma$
  $\longrightarrow$ generates a set of terms $\mathrm{Tm} \in {\mathrm{sort}_\Sigma}^\star \times \mathrm{sort}_\Sigma \to \mathbf{Set}$
- a set of equations $\mathrm{eqs}_\Sigma(v, s) \subseteq \mathrm{Tm}(v, s) \times \mathrm{Tm}(v, s)$

Multi-sorted universal algebra

Monoids, rings, propositional logic... fit in universal algebra
Modules, vector spaces... need multiple sorts

- two sorts = s for scalars and v for vectors
- operations $\mathrm{ar}(\cdot_v) := ([s, v], v)$ $\mathrm{ar}(\cdot_s) := ([s, s], s)$ $\mathrm{ar}(+_s) := ([s, s], s)$ $\mathrm{ar}(+_v) := ([v, v], v)$
- + equations

Multi-sorted signature

A signature Σ consists of:

- a **set of sorts** $\mathrm{sort}_\Sigma \in \mathbf{Set}$
- a set of operations op_Σ, specified by an arity $\mathrm{ar}_\Sigma \in \mathrm{op}_\Sigma \to {\mathrm{sort}_\Sigma}^\star \times \mathrm{sort}_\Sigma$
  $\longrightarrow$ generates a set of terms $\mathrm{Tm} \in {\mathrm{sort}_\Sigma}^\star \times \mathrm{sort}_\Sigma \to \mathbf{Set}$
- a set of equations $\mathrm{eqs}_\Sigma(v, s) \subseteq \mathrm{Tm}(v, s) \times \mathrm{Tm}(v, s)$

Still not enough!

But there's more!

- Category: set of morphisms $\mathrm{Hom}(X, Y)$ for objects X, Y
- First-order logic: expressions, formulas over a given context
- Type theory: types over a context, terms of a type in a context

Still not enough!

But there's more!

- Category: set of morphisms $\mathrm{Hom}(X, Y)$ for objects X, Y
- First-order logic: expressions, formulas over a given context
- Type theory: types over a context, terms of a type in a context

Missing ingredient = dependent sorts

More precisely

A category:

- sorts $\mathrm{ob} \in \mathbf{Set}$ and $\mathrm{hom} \in \prod_{x,y \in \mathrm{ob}} \mathbf{Set}$
- operations $\mathrm{id} \in \prod_{x \in \mathrm{ob}} \mathrm{hom}(x,x)$ and $\circ \in \prod_{g \in \mathrm{hom}(y,z), f \in \mathrm{hom}(x,y)} \mathrm{hom}(x,z)$
- equations $\prod_{f \in \mathrm{hom}(x,y)} f \circ \mathrm{id}(x) = f$ etc.

More precisely

A category:

- sorts $\mathsf{ob} \in \mathbf{Set}$ and $\mathsf{hom} \in \prod_{x,y \in \mathsf{ob}} \mathbf{Set}$
- operations $\mathsf{id} \in \prod_{x \in \mathsf{ob}} \mathsf{hom}(x,x)$ and $\circ \in \prod_{g \in \mathsf{hom}(y,z), f \in \mathsf{hom}(x,y)} \mathsf{hom}(x,z)$
- equations $\prod_{f \in \mathsf{hom}(x,y)} f \circ \mathsf{id}(x) = f$ etc.

A (model of) dependent type theory:

- $\mathsf{Ctx} \in \mathbf{Set}$ (contexts)
- $\mathsf{Ty} \in \mathsf{Ctx} \to \mathbf{Set}$ (types)
- $\mathsf{Tm} \in \prod_{\Gamma \in \mathsf{Ctx}} \mathsf{Ty}(\Gamma) \to \mathbf{Set}$ (terms)
- ...

Use a type theory (= the logical framework) to define signatures

Use a type theory (= the logical framework) to define signatures

Signature, take 2

A signature is a context in the type theory below.

$$\frac{}{\Sigma \vdash \mathsf{Sort}} \qquad \frac{\Sigma \vdash X : \mathsf{Sort}}{\Sigma \vdash \mathsf{El}(X)}$$

$$\frac{\Sigma \vdash A : \mathsf{Sort} \qquad \Sigma \vdash l : \mathsf{El}(A) \qquad \Sigma \vdash r : \mathsf{El}(A)}{\Sigma \vdash l \equiv_A r} \qquad \text{(extensional)}$$

$$\Sigma, (X\colon \mathsf{Sort}) \qquad \Sigma', (x\colon \mathsf{El}(X)) \qquad \Sigma'', (e\colon x \equiv_A y)$$

Use a type theory (= the logical framework) to define signatures

Signature, take 2

A signature is a context in the type theory below.

$$\frac{}{\Sigma \vdash \mathsf{Sort}} \qquad \frac{\Sigma \vdash X : \mathsf{Sort}}{\Sigma \vdash \mathsf{El}(X)}$$

$$\frac{\Sigma \vdash A : \mathsf{Sort} \qquad \Sigma \vdash l : \mathsf{El}(A) \qquad \Sigma \vdash r : \mathsf{El}(A)}{\Sigma \vdash l \equiv_A r} \quad \text{(extensional)}$$

$$\frac{\Sigma \vdash A : \mathsf{Sort} \qquad \Sigma, (x{:}\,A) \vdash B}{\vdash (x{:}\,A) \to B} \qquad \frac{\Sigma \vdash f : (x{:}\,A) \to B \qquad \Sigma \vdash u : \mathsf{El}(A)}{\Sigma \vdash f\ u : \mathsf{El}(B[u])}$$

$$\Sigma, (X : \mathsf{Sort}) \qquad \Sigma', (f : \mathsf{El}(X) \to \mathsf{El}(X)) \qquad \Sigma'', (e : (x{:}\,\mathsf{El}(X)) \to f\ (f\ x) \equiv_X x)$$

Use a type theory (= the logical framework) to define signatures

Signature, take 2

A signature is a context in the type theory below.

$$\frac{}{\Sigma \vdash \mathsf{Sort}} \qquad \frac{\Sigma \vdash X : \mathsf{Sort}}{\Sigma \vdash \mathsf{El}(X)}$$

$$\frac{\Sigma \vdash A : \mathsf{Sort} \qquad \Sigma \vdash l : \mathsf{El}(A) \qquad \Sigma \vdash r : \mathsf{El}(A)}{\Sigma \vdash l \equiv_A r} \qquad \text{(extensional)}$$

$$\frac{\Sigma \vdash A : \mathsf{Sort} \qquad \Sigma, (x\colon A) \vdash B}{\vdash (x\colon A) \to B} \qquad \frac{\Sigma \vdash f : (x\colon A) \to B \qquad \Sigma \vdash u : \mathsf{El}(A)}{\Sigma \vdash f\ u : \mathsf{El}(B[u])}$$

For now: work with this type theory informally

The GAT of categories, very formally

$$
\begin{aligned}
\mathsf{ob} &: \mathsf{Sort} \\
\mathsf{hom} &: (x : \mathsf{ob}) \to (y : \mathsf{ob}) \to \mathsf{Sort} \\
\\
\mathsf{id} &: (x : \mathsf{ob}) \to \mathsf{El}(\mathsf{hom}(x, x)) \\
\mathsf{comp} &: (x, y, z : \mathsf{ob}) \to (f : \mathsf{hom}(x, y)) \to (g : \mathsf{hom}(y, z)) \to \mathsf{El}(\mathsf{hom}(x, z)) \\
\\
\mathsf{idl} &: (x, y : \mathsf{ob}) \to (f : \mathsf{hom}(x, y)) \to \mathsf{comp}(x, x, y, \mathsf{id}(x), f) \equiv_{\mathsf{hom}(x,y)} f \\
\mathsf{idr} &: (x, y : \mathsf{ob}) \to (f : \mathsf{hom}(x, y)) \to \mathsf{comp}(x, y, y, f, \mathsf{id}(y)) \equiv_{\mathsf{hom}(x,y)} f \\
\mathsf{assoc} &: (w, x, y, z : \mathsf{ob}) \to (f : \mathsf{hom}(w, x)) \to (g : \mathsf{hom}(x, y)) \to (h : \mathsf{hom}(y, z)) \to \\
&\qquad \mathsf{comp}(w, x, z, f, \mathsf{comp}(x, y, z, g, h)) \equiv_{\mathsf{hom}(w,z)} \mathsf{comp}(w, y, z\, \mathsf{comp}(w, x, y, f, g), h)
\end{aligned}
$$

The GAT of categories, very formally

$$
\begin{aligned}
\mathsf{ob} &: \mathsf{Sort} \\
\mathsf{hom} &: (x : \mathsf{ob}) \to (y : \mathsf{ob}) \to \mathsf{Sort} \\
\\
\mathsf{id} &: (x : \mathsf{ob}) \to \mathsf{El}(\mathsf{hom}(x, x)) \\
\mathsf{comp} &: (x, y, z : \mathsf{ob}) \to (f : \mathsf{hom}(x, y)) \to (g : \mathsf{hom}(y, z)) \to \mathsf{El}(\mathsf{hom}(x, z)) \\
\\
\mathsf{idl} &: (x, y : \mathsf{ob}) \to (f : \mathsf{hom}(x, y)) \to \mathsf{comp}(x, x, y, \mathsf{id}(x), f) \equiv_{\mathsf{hom}(x,y)} f \\
\mathsf{idr} &: (x, y : \mathsf{ob}) \to (f : \mathsf{hom}(x, y)) \to \mathsf{comp}(x, y, y, f, \mathsf{id}(y)) \equiv_{\mathsf{hom}(x,y)} f \\
\mathsf{assoc} &: (w, x, y, z : \mathsf{ob}) \to (f : \mathsf{hom}(w, x)) \to (g : \mathsf{hom}(x, y)) \to (h : \mathsf{hom}(y, z)) \to \\
&\qquad \mathsf{comp}(w, x, z, f, \mathsf{comp}(x, y, z, g, h)) \equiv_{\mathsf{hom}(w,z)} \mathsf{comp}(w, y, z\, \mathsf{comp}(w, x, y, f, g), h)
\end{aligned}
$$

What a mouthful!

The GAT of categories, very formally

$$
\begin{aligned}
\mathsf{ob} &: \mathsf{Sort} \\
\mathsf{hom} &: (x : \mathsf{ob}) \to (y : \mathsf{ob}) \to \mathsf{Sort} \\
\\
\mathsf{id} &: \{x : \mathsf{ob}\} \to \mathsf{El}(\mathsf{hom}(x, x)) \\
\circ &: \{x, y, z : \mathsf{ob}\} \to (g : \mathsf{hom}(y, z)) \to (f : \mathsf{hom}(x, y)) \to \mathsf{El}(\mathsf{hom}(x, z)) \\
\\
& (x, y : \mathsf{ob}) \to (f : \mathsf{hom}(x, y)) \to f \circ \mathsf{id}_x \equiv f \\
& (x, y : \mathsf{ob}) \to (f : \mathsf{hom}(x, y)) \to \mathsf{id}_y \circ f \equiv f \\
& (w, x, y, z : \mathsf{ob}) \to (f : \mathsf{hom}(w, x)) \to (g : \mathsf{hom}(x, y)) \to (h : \mathsf{hom}(y, z)) \to \\
& \qquad (h \circ g) \circ f \equiv h \circ (g \circ f)
\end{aligned}
$$

What a mouthful! Conventions: implicit arguments + notations

The GAT of categories, very formally

$$\frac{}{\vdash \mathrm{ob}} \qquad \frac{\vdash x : \mathrm{ob} \qquad \vdash y : \mathrm{ob}}{\vdash \mathrm{hom}(x, y)}$$

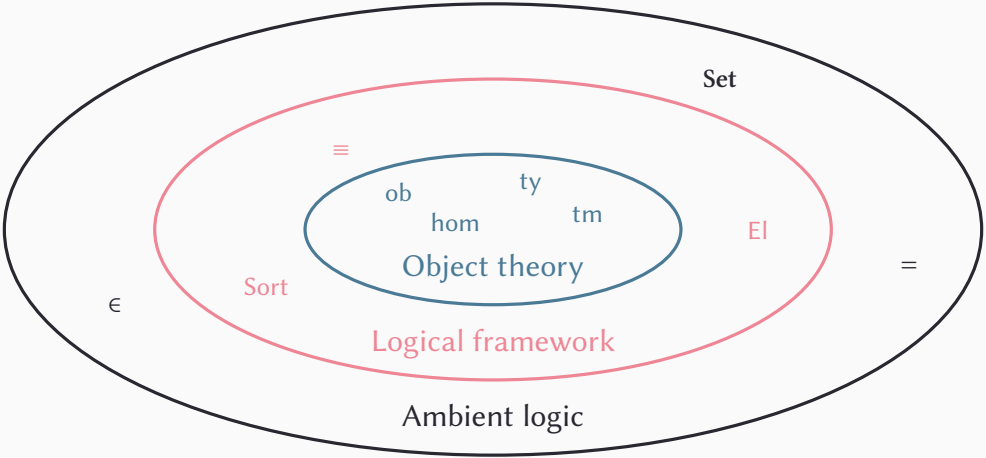
$$\frac{\vdash x : \mathrm{ob}}{\vdash \mathrm{id}_x : \mathrm{hom}(x, y)} \qquad \frac{\vdash x, y, z : \mathrm{ob} \qquad \vdash f : \mathrm{hom}(x, y) \qquad \vdash g : \mathrm{hom}(y, z)}{\vdash g \circ f : \mathrm{hom}(x, z)}$$

$$\frac{\vdash x, y : \mathrm{ob} \qquad \vdash f : \mathrm{hom}(x, y)}{\vdash f \circ \mathrm{id}_x \equiv f : \mathrm{hom}(x, y)} \qquad \frac{\vdash x, y : \mathrm{ob} \qquad \vdash f : \mathrm{hom}(x, y)}{\vdash \mathrm{id}_y \circ f \equiv f : \mathrm{hom}(x, y)}$$

$$\frac{\vdash w, x, y, z : \mathrm{ob} \qquad \vdash f : \mathrm{hom}(w, x) \qquad \vdash g : \mathrm{hom}(x, y) \qquad \vdash h : \mathrm{hom}(y, z)}{\vdash (h \circ g) \circ f \equiv h \circ (g \circ f) : \mathrm{hom}(w, z)}$$

What a mouthful! As typing rules

The type theorist's favourite GAT: Categories with Families



The type theorist's favourite GAT: Categories with Families

$$\begin{aligned} \mathsf{ctx} &: \mathsf{Sort} \\ \mathsf{sub} &: (\Gamma : \mathsf{ctx}) \to (\Delta : \mathsf{ctx}) \to \mathsf{Sort} \\ &\dots \end{aligned}$$

ctx and sub form a category

The type theorist's favourite GAT: Categories with Families

$$
\begin{aligned}
&\mathsf{ctx} : \mathsf{Sort} \\
&\mathsf{sub} : (\Gamma : \mathsf{ctx}) \to (\Delta : \mathsf{ctx}) \to \mathsf{Sort} \\
&\quad \dots \\
&\mathsf{ty} : (\Gamma : \mathsf{ctx}) \to \mathsf{Sort} \\
&\_[\_] : \{\Gamma, \Delta : \mathsf{ctx}\} \to (X : \mathsf{ty}(\Delta)) \to (\sigma : \mathsf{sub}(\Gamma, \Delta)) \to \mathsf{El}(\mathsf{ty}(\Gamma)) \\
&\quad (\Gamma : \mathsf{ctx}) \to (X : \mathsf{ty}(\Gamma)) \to X[\mathsf{id}] \equiv X \\
&\quad \dots
\end{aligned}
$$

ctx and sub form a category

ty is a functor

The type theorist's favourite GAT: Categories with Families

$$
\begin{array}{rll}
\mathsf{ctx} : & \mathsf{Sort} & \\
\mathsf{sub} : & (\Gamma : \mathsf{ctx}) \to (\Delta : \mathsf{ctx}) \to \mathsf{Sort} & \mathsf{ctx} \text{ and } \mathsf{sub} \text{ form a category} \\
& \dots & \\
 & & \\
\mathsf{ty} : & (\Gamma : \mathsf{ctx}) \to \mathsf{Sort} & \\
\_[\_] : & \{\Gamma, \Delta : \mathsf{ctx}\} \to (X : \mathsf{ty}(\Delta)) \to (\sigma : \mathsf{sub}(\Gamma, \Delta)) \to \mathsf{El}(\mathsf{ty}(\Gamma)) & \\
 & (\Gamma : \mathsf{ctx}) \to (X : \mathsf{ty}(\Gamma)) \to X[\mathsf{id}] \equiv X & \mathsf{ty} \text{ is a functor} \\
 & \dots & \\
 & & \\
\mathsf{tm} : & (\Gamma : \mathsf{ctx}) \to (X : \mathsf{ty}(\Gamma)) \to \mathsf{Sort} & \\
\_[\_] : & \{\Gamma, \Delta : \mathsf{ctx}\} \to \{X : \mathsf{ty}(\Delta)\} \to (x : \mathsf{tm}(\Delta, X)) \to (\sigma : \mathsf{sub}(\Gamma, \Delta)) \to & \\
 & \quad \mathsf{El}(\mathsf{tm}(\Gamma, X[\sigma])) & \mathsf{tm} \text{ is a displayed functor} \\
 & (\Gamma : \mathsf{ctx}) \to (X : \mathsf{ty}(\Gamma)) \to (x : \mathsf{tm}(\Gamma, X)) \to x[\mathsf{id}] \equiv_{\mathsf{tm}(\Gamma, X)} x & \\
 & \dots &
\end{array}
$$

The type theorist's favourite GAT: Categories with Families

$$\begin{aligned}
&\mathsf{ctx} : \mathsf{Sort} \\
&\mathsf{sub} : (\Gamma : \mathsf{ctx}) \to (\Delta : \mathsf{ctx}) \to \mathsf{Sort} && \text{ctx and sub form a category} \\
&\dots \\
&\mathsf{ty} : (\Gamma : \mathsf{ctx}) \to \mathsf{Sort} \\
&_[_] : \{\Gamma, \Delta : \mathsf{ctx}\} \to (X : \mathsf{ty}(\Delta)) \to (\sigma : \mathsf{sub}(\Gamma, \Delta)) \to \mathsf{El}(\mathsf{ty}(\Gamma)) \\
&\quad (\Gamma : \mathsf{ctx}) \to (X : \mathsf{ty}(\Gamma)) \to X[\mathsf{id}] \equiv X && \text{ty is a functor} \\
&\quad \dots \\
&\mathsf{tm} : (\Gamma : \mathsf{ctx}) \to (X : \mathsf{ty}(\Gamma)) \to \mathsf{Sort} \\
&_[_] : \{\Gamma, \Delta : \mathsf{c} \\
&\quad \mathsf{El}(\mathsf{tr} && \text{nctor} \\
&\quad (\Gamma : \mathsf{ctx}) \\
&\quad \dots
\end{aligned}$$

$$(\Gamma : \mathsf{ctx}) \to (X : \mathsf{ty}(\Gamma)) \to (x : \mathsf{tm}(\Gamma, X)) \to x[\mathsf{id}] \equiv_{\mathsf{tm}(\Gamma, X)} x$$

The type theorist's favourite GAT: Categories with Families (Ctd.)

$\diamond :\quad \mathsf{El}(\mathsf{ctx})$

$! :\quad \{\Gamma : \mathsf{ctx}\} \to \mathsf{El}(\mathsf{sub}(\Gamma, \diamond))$

$(\Gamma : \mathsf{ctx}) \to (\sigma : \mathsf{sub}(\Gamma, \diamond)) \to \sigma \equiv !$

ctx has a terminal object (empty context)

The type theorist's favourite GAT: Categories with Families (Ctd.)

$\diamond :\quad \mathsf{El}(\mathsf{ctx})$

$! :\quad \{\Gamma : \mathsf{ctx}\} \to \mathsf{El}(\mathsf{sub}(\Gamma, \diamond))$ — ctx has a terminal object (empty context)

$\quad\quad (\Gamma : \mathsf{ctx}) \to (\sigma : \mathsf{sub}(\Gamma, \diamond)) \to \sigma \equiv !$

$\_ \triangleright \_ :\quad (\Gamma : \mathsf{ctx}) \to (X : \mathsf{ty}(\Gamma)) \to \mathsf{El}(\mathsf{ctx})$ — context extension $(\Gamma, x : X)$

The type theorist's favourite GAT: Categories with Families (Ctd.)

$$
\begin{array}{rll}
\diamond : & \mathsf{El}(\mathsf{ctx}) & \\
! : & \{\Gamma : \mathsf{ctx}\} \to \mathsf{El}(\mathsf{sub}(\Gamma, \diamond)) & \text{ctx has a terminal object (empty context)} \\
 & (\Gamma : \mathsf{ctx}) \to (\sigma : \mathsf{sub}(\Gamma, \diamond)) \to \sigma \equiv ! & \\
 & & \\
\_ \triangleright \_ : & (\Gamma : \mathsf{ctx}) \to (X : \mathsf{ty}(\Gamma)) \to \mathsf{El}(\mathsf{ctx}) & \text{context extension } (\Gamma, x : X) \\
\mathsf{var} : & \{\Gamma : \mathsf{ctx}\} \to \{X : \mathsf{ty}(\Gamma)\} \to \mathsf{El}(\mathsf{tm}(\Gamma \triangleright X, X\quad)) & \text{last variable in context}
\end{array}
$$

The type theorist's favourite GAT: Categories with Families (Ctd.)

$\diamond :$	$\mathsf{El}(\mathsf{ctx})$	
$! :$	$\{\Gamma : \mathsf{ctx}\} \to \mathsf{El}(\mathsf{sub}(\Gamma, \diamond))$	ctx has a terminal object (empty context)
	$(\Gamma : \mathsf{ctx}) \to (\sigma : \mathsf{sub}(\Gamma, \diamond)) \to \sigma \equiv !$	
$\_ \triangleright \_ :$	$(\Gamma : \mathsf{ctx}) \to (X : \mathsf{ty}(\Gamma)) \to \mathsf{El}(\mathsf{ctx})$	context extension $(\Gamma, x : X)$
$\mathsf{var} :$	$\{\Gamma : \mathsf{ctx}\} \to \{X : \mathsf{ty}(\Gamma)\} \to \mathsf{El}(\mathsf{tm}(\Gamma \triangleright X, X[\uparrow]))$	last variable in context
$\uparrow :$	$\{\Gamma : \mathsf{ctx}\} \to \{X : \mathsf{ty}(\Gamma)\} \to \mathsf{El}(\mathsf{sub}(\Gamma \triangleright X, \Gamma))$	weakening

The type theorist's favourite GAT: Categories with Families (Ctd.)

$$
\begin{array}{rll}
\diamond : & \mathsf{El}(\mathsf{ctx}) & \\
! : & \{\Gamma : \mathsf{ctx}\} \to \mathsf{El}(\mathsf{sub}(\Gamma, \diamond)) & \mathsf{ctx} \text{ has a terminal object (empty context)} \\
 & (\Gamma : \mathsf{ctx}) \to (\sigma : \mathsf{sub}(\Gamma, \diamond)) \to \sigma \equiv ! & \\
 & & \\
\_ \triangleright \_ : & (\Gamma : \mathsf{ctx}) \to (X : \mathsf{ty}(\Gamma)) \to \mathsf{El}(\mathsf{ctx}) & \text{context extension } (\Gamma, x : X) \\
\mathsf{var} : & \{\Gamma : \mathsf{ctx}\} \to \{X : \mathsf{ty}(\Gamma)\} \to \mathsf{El}(\mathsf{tm}(\Gamma \triangleright X, X[\uparrow])) & \text{last variable in context} \\
\uparrow : & \{\Gamma : \mathsf{ctx}\} \to \{X : \mathsf{ty}(\Gamma)\} \to \mathsf{El}(\mathsf{sub}(\Gamma \triangleright X, \Gamma)) & \text{weakening} \\
(\_, , \_) : & \{\Gamma, \Delta : \mathsf{ctx}\} \to \{X : \mathsf{ty}(\Gamma)\} \to (\sigma : \mathsf{sub}(\Delta, \Gamma)) \to & \text{substitution extension} \\
 & \quad (x : \mathsf{tm}(\Delta, X[\sigma])) \to \mathsf{El}(\mathsf{sub}(\Delta, \Gamma \triangleright X)) &
\end{array}
$$

+ equations

The type theorist's favourite GAT: Categories with Families (Ctd.)

$$
\begin{array}{rll}
\diamond : & \mathrm{El}(\mathsf{ctx}) & \\
! : & \{\Gamma : \mathsf{ctx}\} \to \mathrm{El}(\mathsf{sub}(\Gamma, \diamond)) & \mathsf{ctx} \text{ has a terminal object (empty context)} \\
 & (\Gamma : \mathsf{ctx}) \to (\sigma : \mathsf{sub}(\Gamma, \diamond)) \to \sigma \equiv ! & \\
 & & \\
\_ \triangleright \_ : & (\Gamma : \mathsf{ctx}) \to (X : \mathsf{ty}(\Gamma)) \to \mathrm{El}(\mathsf{ctx}) & \text{context extension } (\Gamma, x : X) \\
\mathsf{var} : & \{\Gamma : \mathsf{ctx}\} \to \{X : \mathsf{ty}(\Gamma)\} \to \mathrm{El}(\mathsf{tm}(\Gamma \triangleright X, X[\uparrow])) & \text{last variable in context} \\
\uparrow : & \{\Gamma : \mathsf{ctx}\} \to \{X : \mathsf{ty}(\Gamma)\} \to \mathrm{El}(\mathsf{sub}(\Gamma \triangleright X, \Gamma)) & \text{weakening} \\
(\_, , \_) : & \{\Gamma, \Delta : \mathsf{ctx}\} \to \{X : \mathsf{ty}(\Gamma)\} \to (\sigma : \mathsf{sub}(\Delta, \Gamma)) \to & \text{substitution extension} \\
 & \quad (x : \mathsf{tm}(\Delta, X[\sigma])) \to \mathrm{El}(\mathsf{sub}(\Delta, \Gamma \triangleright X)) &
\end{array}
$$

+ equations

Only terms: $\mathsf{var}[\uparrow^n]$ = n-th variable in context

Models of a GAT

The logical framework looks a lot like the ambient logic...

The standard model

The logical framework looks a lot like the ambient logic...
We can use that to interpret it! (≈ Tarski semantics...)

The standard model

The logical framework looks a lot like the ambient logic...
We can use that to interpret it! (≈ Tarski semantics...)

$$
\begin{array}{ccc}
 & [\![\_]\!] & \\
\mathsf{Sort} & \mapsto & \mathbf{Set} \\
\mathsf{El}(X) & \mapsto & [\![X]\!] \\
\equiv & \mapsto & = \\
(\_) \to \_ & \mapsto & \prod \\
\_\,,\,\_ & \mapsto & \sum
\end{array}
$$

The standard model

The logical framework looks a lot like the ambient logic...
We can use that to interpret it! (≈ Tarski semantics...)

$$
\begin{array}{ccc}
 & [\![\_]\!] & \\
\mathsf{Sort} & \mapsto & \mathbf{Set} \\
\mathsf{El}(X) & \mapsto & [\![X]\!] \\
\equiv & \mapsto & = \\
(\_) \to \_ & \mapsto & \prod \\
\_, \_ & \mapsto & \sum
\end{array}
$$

Definition

A **model** of a GAT Σ is an inhabitant of $[\![\Sigma]\!]$.

The standard model

The logical framework looks a lot like the ambient logic...
We can use that to interpret it! (≈ Tarski semantics...)

$$
\begin{array}{ccc}
 & [\![\_]\!] & \\
\mathsf{Sort} & \mapsto & \mathbf{Set} \\
\mathsf{El}(X) & \mapsto & [\![X]\!] \\
\equiv & \mapsto & = \\
(\_) \to \_ & \mapsto & \prod \\
\_, \_ & \mapsto & \sum
\end{array}
$$

Definition

A **model** of a GAT Σ is an inhabitant of $[\![\Sigma]\!]$.

In fact, we have a **category** $\mathrm{Mod}(\Sigma)$ (might be too strict to your liking!)

What is a CwF?

Read from the GAT definition! Or...

What is a CwF?

Read from the GAT definition! Or...

Definition (Fiore 2012, Awodey 2018)

A **natural model** consists of:

- a category $\mathscr{C}$ with a terminal object = contexts + substitutions + empty context
- two presheaves Ty and Tm (functors $\mathscr{C}^{\mathrm{op}} \to \mathbf{Set}$) = types, terms, action of substitutions
- a natural transformation $ty : Tm \to Ty$ = maps a term to its type

What is a CwF?

Read from the GAT definition! Or...

Definition (Fiore 2012, Awodey 2018)

A **natural model** consists of:

- a category $\mathscr{C}$ with a terminal object = contexts + substitutions + empty context
- two presheaves Ty and Tm (functors $\mathscr{C}^{\mathrm{op}} \to \mathbf{Set}$) = types, terms, action of substitutions
- a natural transformation $ty : Tm \to Ty$ = maps a term to its type
- such that ty is locally representable = variables

What is a CwF?

Read from the GAT definition! Or...

Definition (Fiore 2012, Awodey 2018)

A **natural model** consists of:

- a category $\mathscr{C}$ with a terminal object = contexts + substitutions + empty context
- two presheaves Ty and Tm (functors $\mathscr{C}^{\mathrm{op}} \to \mathbf{Set}$) = types, terms, action of substitutions
- a natural transformation $ty : Tm \to Ty$ = maps a term to its type
- such that ty is locally representable = variables

"$\Gamma \triangleright A$ is the universal context which extends Γ and has a term of type A"

$$\begin{array}{ccc} \mathrm{y}(\Gamma \triangleright A) & \xrightarrow{\mathrm{var}} & Tm \\ \uparrow \downarrow & \lrcorner & \downarrow ty \\ \mathrm{y}\,\Gamma & \xrightarrow[A]{} & Ty \end{array}$$

What is a CwF?

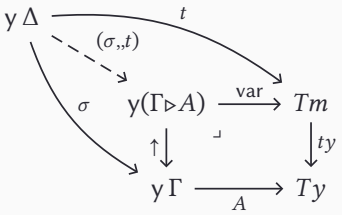
Read from the GAT definition! Or...

Definition (Fiore 2012, Awodey 2018)

A **natural model** consists of:

- a category $\mathscr{C}$ with a terminal object = contexts + substitutions + empty context
- two presheaves Ty and Tm (functors $\mathscr{C}^{\mathrm{op}} \to \mathbf{Set}$) = types, terms, action of substitutions
- a natural transformation $ty : Tm \to Ty$ = maps a term to its type
- such that ty is locally representable = variables

"$\Gamma \triangleright A$ is the universal context which extends Γ and has a term of type A"



What is a CwF?

Read from the GAT defi

Definition (Fiore 2012

A **natural model** consi

- a category $\mathscr{C}$ with
- two presheaves Ty
- a natural transform
- such that ty is loca

"$\Gamma \triangleright A$ is the universal co

mpty context

ction of substitutions

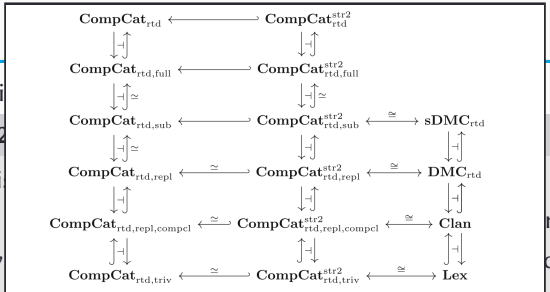


Fig. 4. Notions with types as maps, rooted

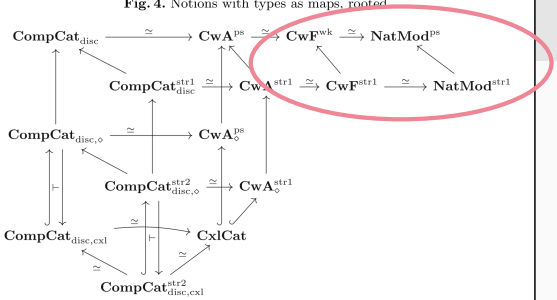


Fig. 5. Notions with types primitive

Syntax

What do I mean by syntax?

Constructions that are valid in every model?

Syntax is the initial model

What do I mean by syntax?

Constructions that are valid in every model?

Syntax is the initial model

Some sort of manipulation of syntax trees?

What do I mean by syntax?

Constructions that are valid in every model?

Syntax is the initial model

Some sort of manipulation of syntax trees?

It is the same!

Why are syntax tree not enough?

Ill-formed syntax trees: $0 + +$ $\lambda\, x.y$

Ill-typed syntax trees: $0 + (\lambda\, x.x)$

$\longrightarrow$ some syntax trees do not represent any valid terms

Why are syntax tree not enough?

Ill-formed syntax trees: $0 + +$ $\lambda\, x.y$

Ill-typed syntax trees: $0 + (\lambda\, x.x)$

$\longrightarrow$ some syntax trees do not represent any valid terms

Equations: $\mathrm{id} \circ f = f$
$\longrightarrow$ many syntax trees represent equal terms

Why are syntax tree not enough?

Ill-formed syntax trees: $0 + +$ $\lambda\, x.y$

Ill-typed syntax trees: $0 + (\lambda\, x.x)$

$\longrightarrow$ some syntax trees do not represent any valid terms

Equations: $\mathrm{id} \circ f = f$
$\longrightarrow$ many syntax trees represent equal terms

If f represents a valid term of type $\hom(a, b)$, is $f \circ \mathrm{id}_{a'}$ valid?
$\longrightarrow$ equations play a role in typing!

Theorem (Cartmell 1986, Kaposi, Kovács and Altenkirch 2019)

The initial model of any GAT exists.

1. Start from syntax trees = “pre-terms”

Theorem (Cartmell 1986, Kaposi, Kovács and Altenkirch 2019)

The initial model of any GAT exists.

1. Start from syntax trees = "pre-terms"
2. (Mutually) define typing and definitional equality of pre-terms (predicates/relations)
   = least relation closed under rules of the theory

Theorem (Cartmell 1986, Kaposi, Kovács and Altenkirch 2019)

The initial model of any GAT exists.

1. Start from syntax trees = "pre-terms"
2. (Mutually) define typing and definitional equality of pre-terms (predicates/relations)
   = least relation closed under rules of the theory
3. Quotient the subset of well-typed pre-terms by definitional equality

Theorem (Cartmell 1986, Kaposi, Kovács and Altenkirch 2019)

The initial model of any GAT exists.

1. Start from syntax trees = "pre-terms"
2. (Mutually) define typing and definitional equality of pre-terms (predicates/relations)
   = least relation closed under rules of the theory
3. Quotient the subset of well-typed pre-terms by definitional equality
4. This is a model, and it is moreover initial!

Theorem (Cartmell 1986, Kaposi, Kovács and Altenkirch 2019)

The initial model of any GAT exists.

1. Start from syntax trees = "pre-terms"
2. (Mutually) define typing and definitional equality of pre-terms (predicates/relations)
   = least relation closed under rules of the theory
3. Quotient the subset of well-typed pre-terms by definitional equality
4. This is a model, and it is moreover initial!

Implementation: work with pre-terms, but check for typing/definitional equality

Much better if typing is **decidable**!

Wrapping up

One type theory to describe them all: a **logical framework**

Do the work **once**, get it for **all theories**:

- syntax
- (category of) models

Syntax is the **initial model**

Where to go from here?

To go on with GATs and CwFs

- *Generalised algebraic theories and contextual categories* (Cartmell 1986)
- *Categories with families: Unityped, simply typed, and dependently typed* (Castellan et al. 2021)
- *Constructing Quotient Inductive-Inductive Types* (Kaposi et al. 2019)
- *On generalized algebraic theories and categories with families* (Bezem et al. 2021)
- *Principles of Dependent Type Theory* (Angiuli and Gratzer 2026) (thanks for the preview!)

Where to go from here?

To go on with GATs and CwFs

- *Generalised algebraic theories and contextual categories* (Cartmell 1986)
- *Categories with families: Unityped, simply typed, and dependently typed* (Castellan et al. 2021)
- *Constructing Quotient Inductive-Inductive Types* (Kaposi et al. 2019)
- *On generalized algebraic theories and categories with families* (Bezem et al. 2021)
- *Principles of Dependent Type Theory* (Angiuli and Gratzer 2026) (thanks for the preview!)

Other logical frameworks

- SOGATs = GATs + built-in handling of contexts & variables
  → *Second-order generalised algebraic theories, by examples* (Kaposi 2025)
- a more syntactic version (// construction of syntax)
  → *A general definition of dependent type theories* (Bauer et al. 2020)
- proof assistants!
  → Isabelle, Dedukti, Andromeda 2, Beluga...

Where to go from here?

To go on with GATs and CwFs

- *Generalised algebraic theories and contextual categories* (Cartmell 1986)
- *Categories with families: Unityped, simply typed, and dependently typed* (Castellan et al. 2021)
- *Constructing Quotient Inductive-Inductive Types* (Kaposi et al. 2019)
- *On generalized algebraic theories and categories with families* (Bezem et al. 2021)
- *Principles of Dependent Type Theory* (Angiuli and Gratzer 2026) (thanks for the preview!)

Other logical frameworks

- SOGATs = GATs + built-in handling of contexts & variables
  → *Second-order generalised algebraic theories, by examples* (Kaposi 2025)
- a more syntactic version (// construction of syntax)
  → *A general definition of dependent type theories* (Bauer et al. 2020)
- proof assistants!
  → Isabelle, Dedukti, Andromeda 2, Beluga...

Theorem (Uemura 2022)

The category of GATs is the bi-initial category with finite limits and an exponentiable morphism.

Thank you!

Bibliography 1

[GTW77] Joseph Goguen, James Thatcher and Eric Wagner. 'An Initial Algebra Approach to the Specification, Correctness and Implementation of Abstract Data Types'. In: *Current Trends in Programming Methodology. Data Structuring*. 1977.

[Car86] John Cartmell. 'Generalised algebraic theories and contextual categories'. In: *Annals of Pure and Applied Logic* (1986). DOI: 10.1016/0168-0072(86)90053-9.

[Fio12] Marcelo Fiore. 'Discrete generalised polynomial functors'. Talk at ICALP 2012. 2012. DOI: 10.1007/978-3-642-31585-5_22.

[Awo18] Steve Awodey. 'Natural models of homotopy type theory'. In: *Mathematical Structures in Computer Science* 2 (2018). DOI: 10.1017/S0960129516000268.

[KKA19] Ambrus Kaposi, András Kovács and Thorsten Altenkirch. 'Constructing Quotient Inductive-Inductive Types'. In: *Proceeding of the ACM on Programming Languages* POPL (Jan. 2019). DOI: 10.1145/3290315.

[BHL20] Andrej Bauer, Philipp G. Haselwarter and Peter LeFanu Lumsdaine. 'A general definition of dependent type theories'. 2020. DOI: 10.48550/arxiv.2009.05539. arXiv: 2009.05539 [math.LO].

[Bez+21] Marc Bezem et al. 'On generalized algebraic theories and categories with families'. In: *Mathematical Structures in Computer Science* 9 (2021). DOI: 10.1017/S0960129521000268.

[CCD21] Simon Castellan, Pierre Clairambault and Peter Dybjer. 'Categories with families: Unityped, simply typed, and dependently typed'. In: *Joachim Lambek: The Interplay of Mathematics, Logic, and Linguistics* (2021).

[Uem22] Taichi Uemura. 'The universal exponentiable arrow'. In: *Journal of Pure and Applied Algebra* 7 (July 2022). DOI: 10.1016/j.jpaa.2021.106991.

[Kap25] Ambrus Kaposi. 'Second-order generalised algebraic theories, by examples'. Lecture notes for the International School on Logical Frameworks and Proof Systems Interoperability. 2025.

Bibliography ii

[Avr+26] Samy Avrillon et al. 'For Generalised Algebraic Theories, Two Sorts Are Enough'. In: (27th Jan. 2026). DOI: 10.48550/ARXIV.2601.19426. arXiv: 2601.19426 [cs.PL].

[AG26] Carlo Angiuli and Daniel Gratzer. *Principles of Dependent Type Theory*. 2026. In preparation.