GAT Signature Languages

A *Begriffsschrift* for Concrete Structures

Jacob Neumann
University of Nottingham
ASSUME - February 2025

Riddle:

How old is Jacob if he's six years older than thrice his age twenty years ago?

Calculemus!

$$
\begin{aligned}
x - 6 &= 3(x - 20) \\
x - 6 &= 3x - 60 \\
x + 54 &= 3x \\
54 &= 2x \\
27 &= x
\end{aligned}
$$

Arithmetic write the problem in the appropriate notation (i.e. equation(s) with unknowns as variable(s)), and *calculate* your answer by applying formulaic rules.

Arithmetic write the problem in the appropriate notation (i.e. equation(s) with unknowns as variable(s)), and *calculate* your answer by applying formulaic rules.

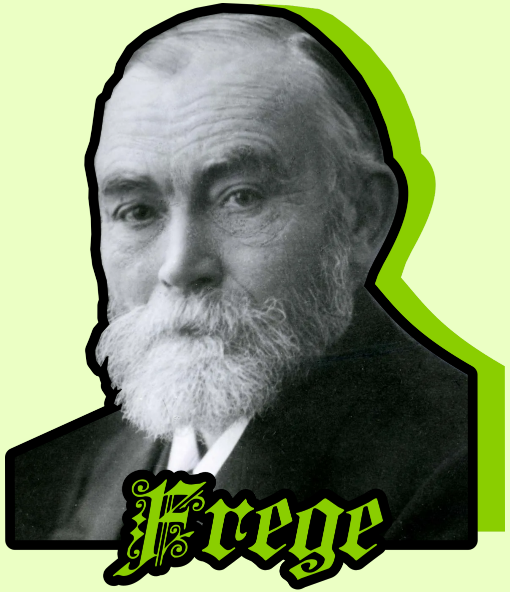
- The true power of the *calculus* is that it can be *calculated with*.
- Calculus write the problem in the appropriate notation ($\int$, d), and *calculate* your answer by applying formulaic rules (e.g. chain rule, (anti)derivatives of polynomials).

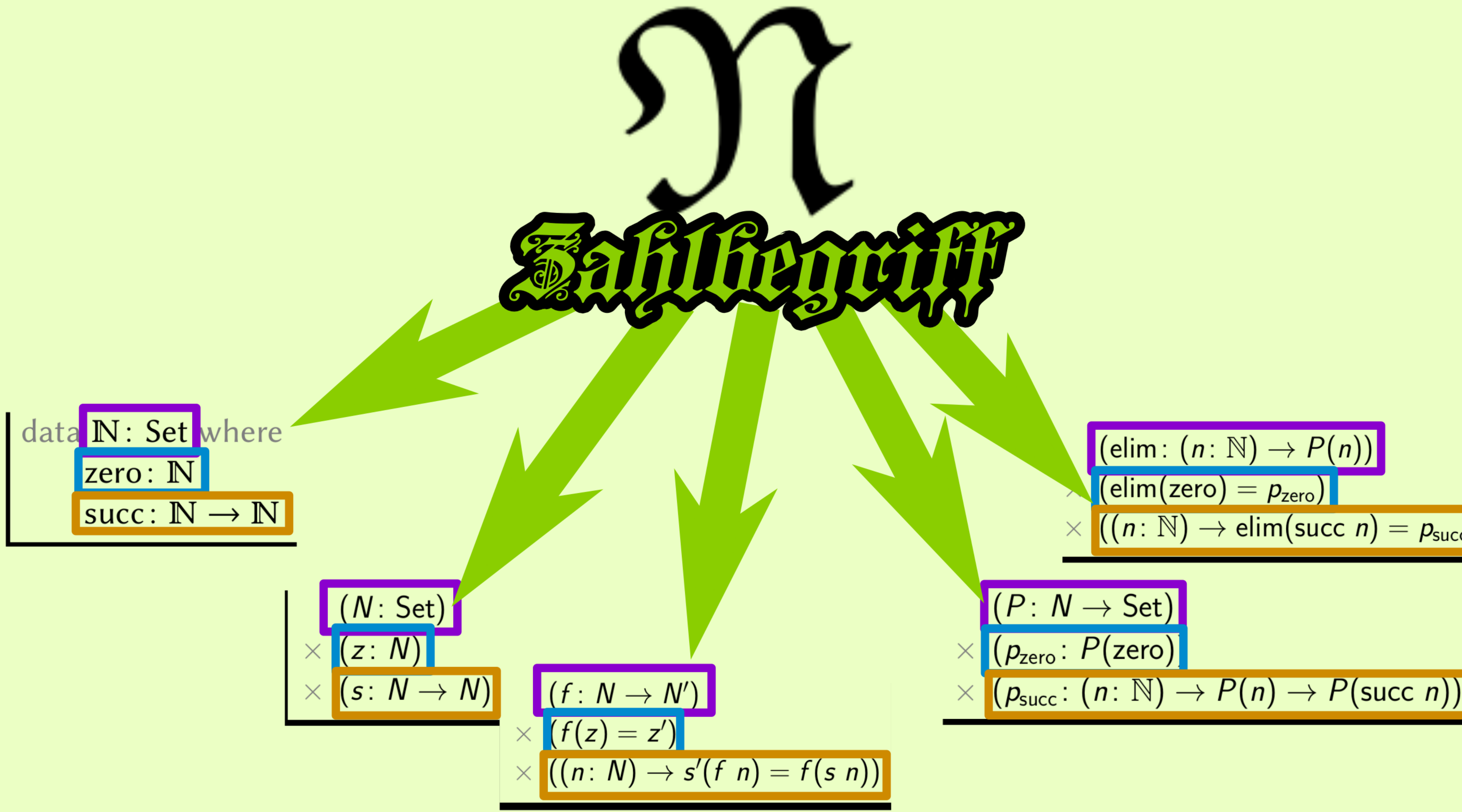
Characteristica Universalis and Begriffsschrift

- What if *everything* could be studied in this way?
- Science/Logic/Metaphysics write down the problem in the appropriate notation (***characteristica universalis***), and calculate your answer by applying formulaic rules (***calculus ratiocinator***).

Though expressing some skepticism about Leibniz's dream, Frege's 1879 ***Begriffsschrift*** partially achieved it, introducing a logic capable of expressing & reasoning about core mathematical objects.

(concept/idea/notion) + (writing/script(ure))





Zahlbegriff
data ℕ : Set where
zero : ℕ
succ : ℕ → ℕ
(N : Set)
× (z : N)
× (s : N → N)
(f : N → N′)
× (f(z) = z′)
× ((n : N) → s′(f n) = f(s n))
(P : N → Set)
× (p_zero : P(zero))
× (p_succ : (n : ℕ) → P(n) → P(succ n))
(elim : (n : ℕ) → P(n))
× (elim(zero) = p_zero)
× ((n : ℕ) → elim(succ n) = p_suc

GATs are a concept-script

- ***Generalized Algebraic Theories*** provide a framework for writing down *Begriffe* like this
- Generalized Algebra write down the concept in the appropriate notation (as a GAT), and *calculate* out how it manifests (as mathematical structures, morphisms, predicates, etc.) by applying formulaic rules
- Not all mathematical structures are GATs, but many of the key ones are
- See the power of generalized algebra in its capacity for self-reflection

Outline

0 Specifying structures as GATs

“As you know, my honourable colleague Mac Lane supports the idea that every structural notion necessarily comes equipped with a notion of homomorphism…What on earth does he hope to deduce from this kind of considerations?”

– Andre Weil, in a letter to Claude Chevalley

“Frightened by the disorder of the discussions, some members had brought a world-renowned efficiency expert from Chicago. This one, armed with a hammer, tried hard and with good humor, but without much result. He quickly realized that it was useless, and turned, successfully this time, to photography.”

– *La Tribu* 34 (1954)

Effrayés du désordre des discussions, certains membres avaient
fait venir de Chicago un "efficiency expert" de renommée mondiale.
Celui-ci, armé d'un marteau, s'évertua avec bonne humeur mais sans
grand résultat. Il comprit vite que c'était inutile, et se tourna,
avec succès cette fois, vers la photographie.

"As you know, my honourable colleague Mac Lane supports the idea that ***every structural notion necessarily comes equipped with a notion of homomorphism***...What on earth does he hope to deduce from this kind of considerations?"

– Andre Weil, in a letter to Claude Chevalley

Central Dogma of Category Theory

Every notion of "structure" comes equipped with a notion of "structure-preserving morphism"

Nat GATs

```
def 𝔑 : GAT := ⦃
    Nat  : U,
    zero : Nat,
    succ : Nat ⇒ Nat
⦄
```

$$\mathfrak{N}\text{ - Alg} = (N : \mathsf{Set}) \times (z : N) \times (s : N \to N)$$

$$(N, z, s) \to (N', z', s') = (f : N \to N') \times (f(z) = z') \times ((n : N) \to s'(f\ n) = f(s\ n))$$

Even/Odd

```
def 𝔈𝔒 : GAT := ⦃
    Even  : U,
    Odd   : U,
    zero  : Even,
    succ  : Even ⇒ Odd,
    succ' : Odd ⇒ Even
⦄
```

$$
\begin{aligned}
\mathfrak{EO}\text{-}\mathsf{Alg} = \\
& (E : \mathsf{Set}) \\
\times\ & (O : \mathsf{Set}) \\
\times\ & (z : E) \\
\times\ & (s : E \to O) \\
\times\ & (s' : O \to E)
\end{aligned}
$$

$$
\begin{aligned}
(E, O, z, s, s') \to (F, P, y, q, q') = \\
& (f : E \to F) \\
\times\ & (g : O \to P) \\
\times\ & (f(z) = y) \\
\times\ & ((e : E) \to q(f\ e) = g(s\ e)) \\
\times\ & ((o : O) \to q'(g\ o) = f(s'\ o))
\end{aligned}
$$

Reflexive Quivers

```
def 𝔯𝔔𝔲𝔦𝔳 : GAT := ⦃
      V : U,
      E : V ⇒ V ⇒ U,
      r : (v : V) ⇒ E v v
⦄
```

$$\mathfrak{rQuiv}\text{-}\mathsf{Alg} = (V : \mathsf{Set}) \times (E : V \to V \to \mathsf{Set}) \times ((v : V) \to E(v, v))$$

$$(V, E, r) \to (V', E', r') = (F_0 : V : V \to V') \times (F_1 : (v_0\ v_1 : V) \to E(v_0, v_1) \to E'(f(v_0), f(v_1))) \times ((v : V) \to r'(F_0\ v) = F_1(r\ v))$$

Algebraic structures

```
def 𝔊𝔯𝔭 : GAT := ⦃
    M      : U,
    u      : M,
    m      : M ⇒ M ⇒ M,
    lunit  : (x : M) ⇒ m u x ≡ x,
    runit  : (x : M) ⇒ m x u ≡ x,
    assoc  : (x : M) ⇒ (y : M) ⇒ (z : M) ⇒
             m x (m y z) ≡ m (m x y) z,
    inv    : M ⇒ M,
    linv   : (x : M) ⇒ m (inv x) x ≡ u,
    rinv   : (x : M) ⇒ m x (inv x) ≡ u
⦄
```

Preserve number of components, even when we don't have to

$$
\begin{aligned}
&(M, u, \mu, \_, \_, \_, i, \_, \_) \to (N, v, \nu, \_, \_, \_, j, \_, \_) = \\
&\quad (\varphi \colon M \to N) \\
&\times\ (\varphi(u) = v) \\
&\times\ ((m_0\ m_1 \colon M) \to \nu(\varphi(m_0), \varphi(m_1)) = \varphi(\mu(m_0, m_1))) \\
&\times\ \top \\
&\times\ \top \\
&\times\ \top \\
&\times\ ((m \colon M) \to j(\varphi\ m) = \varphi(i\ m)) \\
&\times\ \top \\
&\times\ \top
\end{aligned}
$$

And so on...

```
def 𝔓𝔯𝔢𝔒𝔯𝔡 : GAT := ⦃
    X : U,
    leq : X ⇒ X ⇒ U,
    leq-prop : {x x' : X} ⇒ {p q : leq x x'} ⇒ p ≡ q,
    rfl : (x : X) ⇒ leq x x
    trns : {x y z : X} ⇒ leq x y ⇒ leq y z ⇒ leq x z
⦄
```

And so on...

```
def 𝔖𝔢𝔱𝔬𝔦𝔡 : GAT := ⦃
      X : U,
      eq : X ⇒ X ⇒ U,
      eq-prop : {x x' : X} ⇒ {p q : eq x x'} ⇒ p ≡ q,
      rfl : (x : X) ⇒ eq x x
      trns : {x y z : X} ⇒ eq x y ⇒ eq y z ⇒ eq x z
      sym : {x y : X} ⇒ eq x y ⇒ eq y x
⦄
```

And so on...

```
def ℭ𝔞𝔱 := ⦃
    Obj  : U,
    Hom  : Obj ⇒ Obj ⇒ U,
    id   : (X : Obj) ⇒ Hom X X,
    comp  : {X :Obj} ⇒ {Y : Obj} ⇒ {Z : Obj} ⇒
            Hom Y Z ⇒ Hom X Y ⇒ Hom X Z,
    lunit : {X : Obj} ⇒ {Y : Obj} ⇒ (f : Hom X Y) ⇒
            comp (id Y) f ≡ f,
    runit : {X : Obj} ⇒ {Y : Obj} ⇒ (f : Hom X Y) ⇒
            comp f (id X) ≡ f,
    assoc : {W:Obj} ⇒ {X:Obj} ⇒ {Y:Obj} ⇒ {Z:Obj} ⇒
            (e : Hom W X) ⇒ (f : Hom X Y) ⇒
            (g : Hom Y Z) ⇒
            comp g (comp f e) ≡ comp (comp g f) e
⦄
```

```
def 𝔊𝔯𝔭𝔡 := ⦃
    Obj : U,
    Hom : Obj ⇒ Obj ⇒ U,
    id  : (X : Obj) ⇒ Hom X X,
    comp  : {X :Obj} ⇒ {Y : Obj} ⇒ {Z : Obj} ⇒
            Hom Y Z ⇒ Hom X Y ⇒ Hom X Z,
    lunit : {X : Obj} ⇒ {Y : Obj} ⇒ (f : Hom X Y) ⇒
            comp (id Y) f ≡ f,
    runit : {X : Obj} ⇒ {Y : Obj} ⇒ (f : Hom X Y) ⇒
            comp f (id X) ≡ f,
    assoc : {W:Obj} ⇒ {X:Obj} ⇒ {Y:Obj} ⇒ {Z:Obj} ⇒
            (e : Hom W X) ⇒ (f : Hom X Y) ⇒
            (g : Hom Y Z) ⇒
            comp g (comp f e) ≡ comp (comp g f) e,
    inv : (X:Obj) ⇒ (Y:Obj) ⇒ Mor X Y ⇒ Mor Y X,
    linv :  {X : Obj} ⇒ {Y : Obj} ⇒ (f : Hom X Y) ⇒
            comp (inv f) f ≡ id X,
    rinv :  {X : Obj} ⇒ {Y : Obj} ⇒ (f : Hom X Y) ⇒
```

Not allowed!

- ✗ Functions with large domain:
  - ✗ $\mathbf{U} \Rightarrow \mathbf{U}$
  - ✗ $(\mathtt{A} \Rightarrow \mathtt{B}) \Rightarrow \mathtt{C}$
- ✗ Sort equations (NB: Cartmell allows them)

$$\mathtt{X} \equiv \mathtt{Y} \qquad \text{where } X, Y : \mathbf{U}.$$

Theorem (Kaposi-Kovács-Altenkirch, '19) Every GAT has an initial algebra.

Proof Konzept:

- Understand the GAT as a context, and consider terms-in-context:

$$\{\!\![\mathtt{Nat} : \mathbf{U}, \mathtt{zero} : \mathtt{Nat}, \mathtt{succ} : \mathtt{Nat} \Rightarrow \mathtt{Nat}]\!\!\} \vdash \mathtt{t} : \mathtt{Nat}$$

- Construct the initial algebra as the "term model": the set interpreting $\mathtt{Nat}$ is the set of terms of type $\mathtt{Nat}$, the interpretation of $\mathtt{zero}$ is itself, etc.
- Prove initiality.

□

GAT of CwFs

```
def ℭ𝔴𝔉 := {|
      Con  : U,
      Sub  : Con ⇒ Con ⇒ U,
      id   : {Γ : Con} ⇒ Sub Γ Γ,
      comp : {Θ :Con} ⇒ {Δ : Con} ⇒ {Γ : Con} ⇒
           Sub Δ Γ ⇒ Sub Θ Δ ⇒ Sub Θ Γ,
      lunit : {Δ : Con} ⇒ {Γ : Con} ⇒ {γ : Sub Δ Γ} ⇒
           comp (id Γ) γ ≡ γ,
      runit : {Δ : Con} ⇒ {Γ : Con} ⇒ {γ : Sub Δ Γ} ⇒
           comp γ (id Δ) ≡ γ,
      assoc : {Ξ : Con} ⇒ {Θ : Con} ⇒
           {Δ : Con} ⇒ {Γ : Con} ⇒
           (ϑ : Sub Ξ Θ) ⇒ (δ : Sub Θ Δ) ⇒
           (γ : Sub Δ Γ) ⇒
           comp γ (comp ϑ δ) ≡ comp (comp δ γ) ϑ,
```

GAT of CwFs

```
empty : Con,
ε : (Γ : Con) ⇒ Sub Γ empty,
ηε : {Γ : Con} ⇒ (f : Sub Γ empty) ⇒ f ≡ (ε Γ),
Ty : Con ⇒ U,
substTy : {Δ : Con} ⇒ {Γ : Con} ⇒
    Sub Δ Γ ⇒ Ty Γ ⇒ Ty Δ,
idTy : {Γ : Con} ⇒ (A : Ty Γ) ⇒
    substTy (id Γ) A ≡ A,
compTy : {Θ:Con} ⇒ {Δ : Con} ⇒ {Γ : Con} ⇒
    (A : Ty Γ) ⇒ (δ : Sub Θ Δ) ⇒ (γ : Sub Δ Γ) ⇒
    substTy γ (substTy δ A) ≡ substTy (comp γ δ) A,
```

GAT of CwFs

```
Tm : (Γ : Con) ⇒ Ty Γ ⇒ U,
substTm : {Δ : Con} ⇒ {Γ : Con} ⇒ {A : Ty Γ} ⇒
    (γ : Sub Δ Γ) ⇒ Tm Γ A ⇒
    Tm Δ (substTy γ A),
idTm : {Γ : Con} ⇒ {A : Ty Γ} ⇒ (t : Tm Γ A) ⇒
    substTm (id Γ) t ≡ t,
compTm : {Θ : Con} ⇒ {Δ : Con} ⇒ {Γ : Con} ⇒
    {A : Ty Γ} ⇒ (t : Tm Γ A) ⇒
    (δ : Sub Θ Δ) ⇒ (γ : Sub Δ Γ) ⇒
    substTm γ (substTm δ t)          #⟨compTy A γ δ⟩
    ≡ substTm (comp γ δ) t,
```

GAT of CwFs

```
ext : (Γ : Con) ⇒ Ty Γ ⇒ Con,
pair : {Δ : Con} ⇒ {Γ : Con} ⇒ {A : Ty Γ} ⇒
    (γ : Sub Δ Γ) ⇒ Tm Δ (substTy γ A) ⇒
    Sub Δ (ext Γ A),
pair_nat: {Θ : Con} ⇒ {Δ : Con} ⇒ {Γ : Con} ⇒
    {A : Ty Γ} ⇒ (γ : Sub Δ Γ) ⇒
    (t : Tm Δ (substTy γ A)) ⇒ (δ : Sub Θ Δ) ⇒
    comp (pair γ t) δ
    ≡ pair (comp γ δ) (substTm δ t  #⟨compTy A γ δ⟩),
```

```
    p : {Γ : Con} ⇒ (A : Ty Γ) ⇒ Sub (ext Γ A) Γ,
    v : {Γ : Con} ⇒ (A : Ty Γ) ⇒
        Tm (ext Γ A) (substTy (p A) A),
    ext_β₁ : {Δ : Con} ⇒ {Γ : Con} ⇒ {A : Ty Γ} ⇒
        (γ : Sub Δ Γ) ⇒ (t : Tm Δ (substTy γ A)) ⇒
        comp (p A) (pair γ t) ≡ γ,
    ext_β₂ : {Δ : Con} ⇒ {Γ : Con} ⇒ {A : Ty Γ} ⇒
        (γ : Sub Δ Γ) ⇒ (t : Tm Δ (substTy γ A)) ⇒
        substTm (pair γ t) (v A)
                #⟨compTy A (p A) (pair γ t); ext_β₁ γ t⟩
        ≡ t,
    ext_η : (Γ : Con) ⇒ (A : Ty Γ) ⇒
        pair (p Γ A) (v Γ A) ≡ id (ext Γ A)
⟧
```

Idea: Every GAT extension of the GAT of CwFs has an initial algebra

1 The GAT signature language

Idea: Make the above constructions precise

- (Quotient inductive-) Inductively define the type of GATs
- Compile the above syntax down to the precise type
- Make definitions (like algebra and homomorphism)

Contexts (GATs)

$$\diamond : \overline{\mathsf{Con}} \qquad \_ \triangleright \_ : (\mathfrak{G} : \overline{\mathsf{Con}}) \to \overline{\mathsf{Ty}}\ \mathfrak{G} \to \overline{\mathsf{Con}}$$

Variables & Substitution

$$\mathsf{wk} : \overline{\mathsf{Sub}}\ (\mathfrak{G} \triangleright \mathcal{X})\ \mathfrak{G}$$

$$0 : \overline{\mathsf{Tm}}(\mathfrak{G} \triangleright \mathcal{X}, \mathcal{X}[\mathsf{wk}]\mathsf{T}) \qquad n + 1 := n[\mathsf{wk}]\mathsf{t}$$

Universe of Sorts

$$\mathsf{U} \colon \overline{\mathsf{Ty}}\ \mathfrak{G} \qquad \mathsf{El} \colon \overline{\mathsf{Tm}}(\mathfrak{G}, \mathsf{U}) \to \overline{\mathsf{Ty}}\ \mathfrak{G}$$

Pi-types with small domain

$$\Pi \colon (\mathcal{A} \colon \overline{\mathsf{Tm}}(\mathfrak{G}, \mathsf{U})) \to \overline{\mathsf{Ty}}(\mathfrak{G} \triangleright \mathsf{El}\ \mathcal{A}) \to \overline{\mathsf{Ty}}\ \mathfrak{G}$$

$$\_@\_ \colon \overline{\mathsf{Tm}}(\mathfrak{G}, \Pi(\mathcal{A}, \mathcal{B})) \to (a \colon \overline{\mathsf{Tm}}(\mathfrak{G}, \mathsf{El}\ \mathcal{A})) \to \overline{\mathsf{Tm}}(\mathfrak{G}, \mathcal{B}[\mathsf{id}, a]\mathsf{T})$$

Note: No need for λ-abstraction!

Example: Nat

```
def 𝔑 : GAT := ⦃
     Nat   : U,
     zero  : Nat,
     succ  : Nat ⇒ Nat
⦄
```

$\diamond$

$\triangleright\ \mathsf{U}$

$\triangleright\ \mathsf{El}\ 0$

$\triangleright\ \Pi\ 1\ (\mathsf{El}\ 2)$

Example: Even-Odd

```
def 𝔈𝔒 : GAT := ⦃
      Even  : U,
      Odd   : U,
      zero  : Even,
      succ  : Even ⇒ Odd,
      succ' : Odd ⇒ Even
⦄
```

$\diamond$
$\triangleright\ \mathsf{U}$
$\triangleright\ \mathsf{U}$
$\triangleright\ \mathsf{El}\ 1$
$\triangleright\ \Pi\ 2\ (\mathsf{El}\ 2)$
$\triangleright\ \Pi\ 2\ (\mathsf{El}\ 4)$

Extensional Identity Types

$$\mathsf{Eq} \colon \{\mathcal{A} \colon \overline{\mathsf{Tm}}(\mathfrak{G}, \mathsf{U})\} \to \overline{\mathsf{Tm}}(\mathfrak{G}, \mathsf{El}\,\mathcal{A}) \to \overline{\mathsf{Tm}}(\mathfrak{G}, \mathsf{El}\,\mathcal{A}) \to \overline{\mathsf{Ty}}\,\mathfrak{G}$$

(get transport from metatheory by reflection)

GAT machine code

$$
\begin{aligned}
&\diamond \\
&\triangleright\ \mathsf{U} \\
&\triangleright\ \mathsf{El}\ 0 \\
&\triangleright\ \Pi\ 1\ (\Pi\ 2\ (\mathsf{El}\ 3)) \\
&\triangleright\ \Pi\ 2\ (\mathsf{Eq}\ (1\ @\ 2\ @\ 0)\ 0) \\
&\triangleright\ \Pi\ 3\ (\mathsf{Eq}\ (2\ @\ 0\ @\ 3)\ 0) \\
&\triangleright\ \Pi\ 4\ (\Pi\ 5\ (\Pi\ 6\ (\mathsf{Eq}\ (5\ @\ 2\ @\ (5\ @\ 1\ @\ 0))\ (5\ @\ (5\ @\ 2\ @\ 1)\ @\ 0))))
\end{aligned}
$$

GAT machine code

$$
\begin{aligned}
&\diamond \\
&\triangleright\ \mathsf{U} \\
&\triangleright\ \Pi\ 0\ (\Pi\ 1\ \mathsf{U}) \\
&\triangleright\ \Pi\ 1\ (\Pi\ 2\ (\Pi\ (2\ @\ 1\ @\ 0)\ (\Pi\ (3\ @\ 2\ @\ 1)\ (\mathsf{Eq}\ 1\ 0)))) \\
&\triangleright\ \Pi\ 2\ (\mathsf{El}\ (2\ @\ 0\ @\ 0)) \\
&\triangleright\ \Pi\ 3\ (\Pi\ 4\ (\Pi\ 5\ (\Pi\ (5\ @\ 2\ @\ 1)\ (\Pi\ (6\ @\ 2\ @\ 1)\ (\mathsf{El}\ (7\ @\ 4\ @\ 2))))))
\end{aligned}
$$

GAT machine code

$$
\begin{aligned}
&\diamond \\
&\triangleright\ \mathsf{U} \\
&\triangleright\ \Pi\ 0\ (\Pi\ 1\ \mathsf{U}) \\
&\triangleright\ \Pi\ 1\ (\mathsf{El}\ (1\ @\ 0\ @\ 0)) \\
&\triangleright\ \Pi\ 2\ (\Pi\ 3\ (\Pi\ 4\ (\Pi\ (4\ @\ 1\ @\ 0)\ (\Pi\ (5\ @\ 3\ @\ 2)\ (\mathsf{El}\ (6\ @\ 4\ @\ 2)))))) \\
&\triangleright\ \Pi\ 3\ (\Pi\ 4\ (\Pi\ (4\ @\ 1\ @\ 0)\ (\mathsf{Eq}\ (3\ @\ 2\ @\ 1\ @\ 1\ @\ (4\ @\ 1)\ @\ 0)\ 0))) \\
&\triangleright\ \Pi\ 4\ (\Pi\ 5\ (\Pi\ (5\ @\ 1\ @\ 0)\ (\mathsf{Eq}\ (4\ @\ 2\ @\ 2\ @\ 1\ @\ 0\ @\ (5\ @\ 2))\ 0))) \\
&\triangleright\ \Pi\ 5\ (\Pi\ 6\ (\Pi\ 7\ (\Pi\ 8\ (\Pi\ (8\ @\ 3\ @\ 2)\ (\Pi\ (9\ @\ 3\ @\ 2)\ (\Pi\ (10\ @\ 3\ @\ 2) \\
&\quad (\mathsf{Eq}\ (9\ @\ 6\ @\ 5\ @\ 3\ @\ 0\ @\ (9\ @\ 6\ @\ 5\ @\ 4\ @\ 1\ @\ 2)) \\
&\qquad (9\ @\ 6\ @\ 4\ @\ 3\ @\ (9\ @\ 5\ @\ 4\ @\ 3\ @\ 0\ @\ 1)\ @\ 2) \\
&\quad)))))))
\end{aligned}
$$

GAT machine code

- $\mathsf{El}\ 6$
- $\Pi\ 7\ (\mathsf{El}\ (7\ @\ 0\ @\ 1))$
- $\Pi\ 8\ (\Pi\ (8\ @\ 0\ @\ 2)\ (\mathsf{Eq}\ 0\ (2\ @\ 1)))$
- $\Pi\ 9\ \mathsf{U}$
- $\Pi\ 10\ (\Pi\ 11\ (\Pi\ (11\ @\ 1\ @\ 0)\ (\Pi\ (3\ @\ 1)\ (\mathsf{El}\ (4\ @\ 3)))))$
- $\Pi\ 11\ (\Pi\ (2\ @\ 0)\ (\mathsf{Eq}\ (2\ @\ 1\ @\ 1\ @\ (11\ @\ 1)\ @\ 0)\ 0))$
- $\Pi\ 12\ (\Pi\ 13\ (\Pi\ 14\ (\Pi\ (5\ @\ 0)\ (\Pi\ (15\ @\ 2\ @\ 1)\ (\Pi\ (16\ @\ 4\ @\ 3)$
  $(\mathsf{Eq}\ (7\ @\ 4\ @\ 3\ @\ 1\ @\ (7\ @\ 5\ @\ 4\ @\ 0\ @\ 2))$
  $(7\ @\ 5\ @\ 3\ @\ (15\ @\ 5\ @\ 4\ @\ 3\ @\ 1\ @\ 0)\ @\ 2)$
  $))))))$
- $\Pi\ 13\ (\Pi\ (4\ @\ 0)\ \mathsf{U})$
- $\Pi\ 14\ (\Pi\ 15\ (\Pi\ (6\ @\ 0)\ (\Pi\ (16\ @\ 2\ @\ 1)\ (\Pi\ (4\ @\ 2\ @\ 1)$
  $(\mathsf{El}\ (5\ @\ 4\ @\ (8\ @\ 4\ @\ 3\ @\ 1\ @\ 2)))))))$
- $\Pi\ 15\ (\Pi\ (6\ @\ 0)\ (\Pi\ (3\ @\ 1\ @\ 0)$

GAT machine code

- Π 16 (Π 17 (Π 18 (Π (9 @ 0) (Π (6 @ 1 @ 0) (Π (20 @ 4 @ 3) (Π (21 @ 4 @ 3)
  (Eq (transp (10 @ 6 @ 5 @ 4 @ 3 @ 0 @ 1)
  (8 @ 5 @ 4 @ 3 @ 0 @ (8 @ 6 @ 5 @ (12 @ 5 @ 4 @ 0 @ 3) @ 1 @ 2)))
  (8 @ 6 @ 4 @ 3 @ (20 @ 6 @ 5 @ 4 @ 0 @ 1) @ 2)
  )))))))
- Π 17 (Π (8 @ 0) (El 19))
- Π 18 (Π 19 (Π (10 @ 0) (Π (20 @ 2 @ 1) (Π (8 @ 3 @ (11 @ 3 @ 2 @ 0 @ 1))
  (El (22 @ 4 @ (5 @ 3 @ 2)))))))
- Π 19 (Π 20 (Π 21 (Π (12 @ 0) (Π (22 @ 2 @ 1) (Π (10 @ 3 @ (13 @ 3 @ 2 @ 0
  @ 1))
  (Π (24 @ 5 @ 4)
  (Eq (23 @ 6 @ 5 @ (8 @ 4 @ 3) @ (7 @ 5 @ 4 @ 3 @ 2 @ 1) @ 0)
  (7 @ 6 @ 4 @ 3 @ (23 @ 2 @ 0) @
  (transp (13 @ 6 @ 5 @ 4 @ 3 @ 2 @ 0)

GAT machine code

- ▷ $\Pi\ 20\ (\Pi\ (11 @ 0)\ (\mathrm{El}\ (21 @ (4 @ 1 @ 0) @ 1)))$
- ▷ $\Pi\ 21\ (\Pi\ (12 @ 0)\ (\mathrm{El}\ (9 @ (5 @ 1 @ 0) @ (12 @ (5 @ 1 @ 0) @ 1 @ (2 @ 1 @ 0) @ 0))))$
- ▷ $\Pi\ 22\ (\Pi\ 23\ (\Pi\ (14 @ 0)\ (\Pi\ (24 @ 2 @ 1)\ (\Pi\ (12 @ 3 @ (15 @ 3 @ 2 @ 0 @ 1))$ $(\mathrm{Eq}\ (24 @ 4 @ (9 @ 3 @ 2) @ 3 @ (6 @ 3 @ 2) @ (8 @ 4 @ 3 @ 2 @ 1 @ 0))$ $1)))))$
- ▷ $\Pi\ 23\ (\Pi\ 24\ (\Pi\ (15 @ 0)\ (\Pi\ (25 @ 2 @ 1)\ (\Pi\ (13 @ 3 @ (16 @ 3 @ 2 @ 0 @ 1))$
  $(\mathrm{Eq}\ (\mathrm{transp}\ (5 @ 4 @ 3 @ 2 @ 1 @ 0)$
  $(\mathrm{transp}\ (15 @ 4 @ (10 @ 3 @ 2) @ 3 @ 2 @ (7 @ 3 @ 2) @ (9 @ 4 @ 3 @ 2 @ 1 @ 0))$
  $(13 @ 4 @ (10 @ 3 @ 2) @ (17 @ 4 @ 3 @ 1 @ 2) @ (9 @ 4 @ 3 @ 2 @ 1 @ 0) @ (6 @ 3 @ 2))$
  $))$
  0

github.com/jacobneu/GeneralizedAlgebra

Upshot

Now, the type of GATs is given as a quotient inductive-inductive type,so we can explicitly define constructions like $(\_)\text{-}\mathsf{Alg}$ in a *structural, compositional* way.

bitbucket.org/akaposi/finitaryqiit/raw/master/ap-pendix.pdf

Upshot

Now, ... so we can explicit... ...al way.

bit... ...p-

Syntax

$\Gamma : \mathsf{Con}$

$A : \mathsf{Ty}\,\Gamma$

$\sigma : \mathsf{Sub}\,\Gamma\,\Delta$

$t : \mathsf{Tm}\,\Gamma\,A$

$\cdot : \mathsf{Con}$

$\Gamma \rhd A : \mathsf{Con}$

$(A : \mathsf{Ty}\,\Delta)[\sigma : \mathsf{Sub}\,\Gamma\,\Delta] : \mathsf{Ty}\,\Gamma$

$\mathsf{id} : \mathsf{Sub}\,\Gamma\,\Gamma$

$(\sigma : \mathsf{Sub}\,\Theta\,\Delta) \circ (\delta : \mathsf{Sub}\,\Gamma\,\Theta) : \mathsf{Sub}\,\Gamma\,\Delta$

$\epsilon : \mathsf{Sub}\,\Gamma\,\cdot$

$(\sigma : \mathsf{Sub}\,\Gamma\,\Delta), (t : \mathsf{Tm}\,\Gamma\,(A[\sigma])) : \mathsf{Sub}\,\Gamma\,(\Delta \rhd A)$

$\pi_1\,(\sigma : \mathsf{Sub}\,\Gamma\,(\Delta \rhd A)) : \mathsf{Sub}\,\Gamma\,\Delta$

$\pi_2\,(\sigma : \mathsf{Sub}\,\Gamma\,(\Delta \rhd A)) : \mathsf{Tm}\,\Gamma\,(A[\pi_1\,\sigma])$

$(t : \mathsf{Tm}\,\Delta\,A)[\sigma : \mathsf{Sub}\,\Gamma\,\Delta] : \mathsf{Tm}\,\Gamma\,(A[\sigma])$

$[\mathsf{id}] : A[\mathsf{id}] = A$

$[\circ] : A[\sigma \circ \delta] = A[\sigma][\delta]$

$\mathsf{ass} : (\sigma \circ \delta) \circ \nu = \sigma \circ (\delta \circ \nu)$

$\mathsf{idl} : \mathsf{id} \circ \sigma = \sigma$

$\mathsf{idr} : \sigma \circ \mathsf{id} = \sigma$

$\cdot\eta : \{\sigma : \mathsf{Sub}\,\Gamma\,\cdot\} \to \sigma = \epsilon$

$\rhd\beta_1 : \pi_1\,(\sigma, t) = \sigma$

Algebras

$\Gamma^A : \mathsf{Set}$

$A^A : \Gamma^A \to \mathsf{Set}$

$\sigma^A : \Gamma^A \to \Delta^A$

$t^A : (\gamma : \Gamma^A) \to A^A\,\gamma$

$\cdot^A :\equiv \top$

$(\Gamma \rhd A)^A :\equiv (\gamma : \Gamma^A) \times A^A\,\gamma$

$(A[\sigma])^A\,\gamma :\equiv A^A\,(\sigma^A\,\gamma)$

$\mathsf{id}^A\,\gamma :\equiv \gamma$

$(\sigma \circ \delta)^A\,\gamma :\equiv \sigma^A\,(\delta^A\,\gamma)$

$\epsilon^A\,\gamma :\equiv \mathsf{tt}$

$(\sigma, t)^A\,\gamma :\equiv (\sigma^A\,\gamma, t^A\,\gamma)$

$(\pi_1\,\sigma)^A\,\gamma :\equiv \mathsf{proj}_1\,(\sigma^A\,\gamma)$

$(\pi_2\,\sigma)^A\,\gamma :\equiv \mathsf{proj}_2\,(\sigma^A\,\gamma)$

$(t[\sigma])^A\,\gamma :\equiv t^A\,(\sigma^A\,\gamma)$

$[\mathsf{id}]^A :\equiv \mathsf{refl}$

$[\circ]^A :\equiv \mathsf{refl}$

$\mathsf{ass}^A :\equiv \mathsf{refl}$

$\mathsf{idl}^A :\equiv \mathsf{refl}$

$\mathsf{idr}^A :\equiv \mathsf{refl}$

$\cdot\eta^A :\equiv \mathsf{refl}$

${\rhd\beta_1}^A :\equiv \mathsf{refl}$

${\rhd\beta_2}^A :\equiv \mathsf{refl}$

Assuming $\Omega : \mathsf{Con}$, the **initial Ω-algebra** is given by $\mathsf{con}_\Omega :\equiv \Omega^C\,\mathsf{id}$

$\Gamma^C : \mathsf{Sub}\,\Omega\,\Gamma \to \Gamma^A$

$A^C : (\nu : \mathsf{Sub}\,\Omega\,\Gamma) \to \mathsf{Tm}\,\Omega\,(A[\nu]) \to A^A\,(\Gamma^C\,\nu)$

$\sigma^C : (\nu : \mathsf{Sub}\,\Omega\,\Gamma) \to \Delta^C\,(\sigma \circ \nu) = \sigma^A\,(\Gamma^C\,\nu)$

$t^C : (\nu : \mathsf{Sub}\,\Omega\,\Gamma) \to A^C\,\nu\,(t[\nu]) = t^A\,(\Gamma^C\,\nu)$

$\cdot^C\,\nu :\equiv \mathsf{tt}$

$(\Gamma \rhd A)^C\,\nu :\equiv (\Gamma^C\,(\pi_1\,\nu), A^C\,(\pi_1\,\nu)\,(\pi_2\,\nu))$

$(A[\sigma])^C\,\nu\,t :\equiv \mathsf{tr}_{A^A}\,(\sigma^C\,\nu)\,(A^C\,(\sigma \circ \nu)\,t)$

$\mathsf{id}^C\,\nu : \Gamma^C\,\nu = \Gamma^C\,\nu$

$(\sigma \circ \delta)^C\,\nu : \Delta^C\,(\sigma \circ \delta \circ \nu) \overset{\sigma^C\,(\delta \circ \nu)}{=} \sigma^A\,(\Theta^C\,(\delta \circ \nu)) \overset{\delta^C\,\nu}{=} \sigma^A\,(\delta^A\,(\Gamma^C\,\nu))$

$\epsilon^C\,\nu : \mathsf{tt} = \mathsf{tt}$

$(\sigma, t)^C\,\nu : (\Gamma^C\,(\sigma \circ \nu), A^C\,(\sigma \circ \nu)\,(t[\nu])) \overset{\sigma^C\,\nu, t^C\,\nu}{=} (\sigma^A\,(\Gamma^C\,\nu), t^A\,(\Gamma^C\,\nu))$

$(\pi_1\,\sigma)^C\,\nu : \Delta^C\,(\pi_1\,(\sigma \circ \nu)) \overset{\sigma^C\,\nu}{=} \mathsf{proj}_1\,(\sigma^A\,(\Gamma^C\,\nu))$

$(\pi_2\,\sigma)^C\,\nu : A^C\,(\pi_1\,(\sigma \circ \nu))\,(\pi_2\,(\sigma \circ \nu)) \overset{\sigma^C\,\nu}{=} \mathsf{proj}_2\,(\sigma^A\,(\Gamma^C\,\nu))$

$(t[\sigma])^C\,\nu : A^C\,(\sigma \circ \nu)\,(t[\sigma \circ \nu]) \overset{t^C\,(\sigma \circ \nu)}{=} t^A\,(\delta^C\,(\sigma \circ \nu)) \overset{\sigma^C\,\nu}{=} t^A\,(\sigma^A\,(\Gamma^C\,\nu))$

$[\mathsf{id}]^C : A^C\,\nu\,t = A^C\,\nu\,t$

$[\circ]^C : A^C\,(\sigma \circ \delta \circ \nu)\,t = A^C\,(\sigma \circ \delta \circ \nu)\,t$

$\mathsf{ass}^C :\equiv \mathsf{UIP}$

$\mathsf{idl}^C :\equiv \mathsf{UIP}$

$\mathsf{idr}^C :\equiv \mathsf{UIP}$

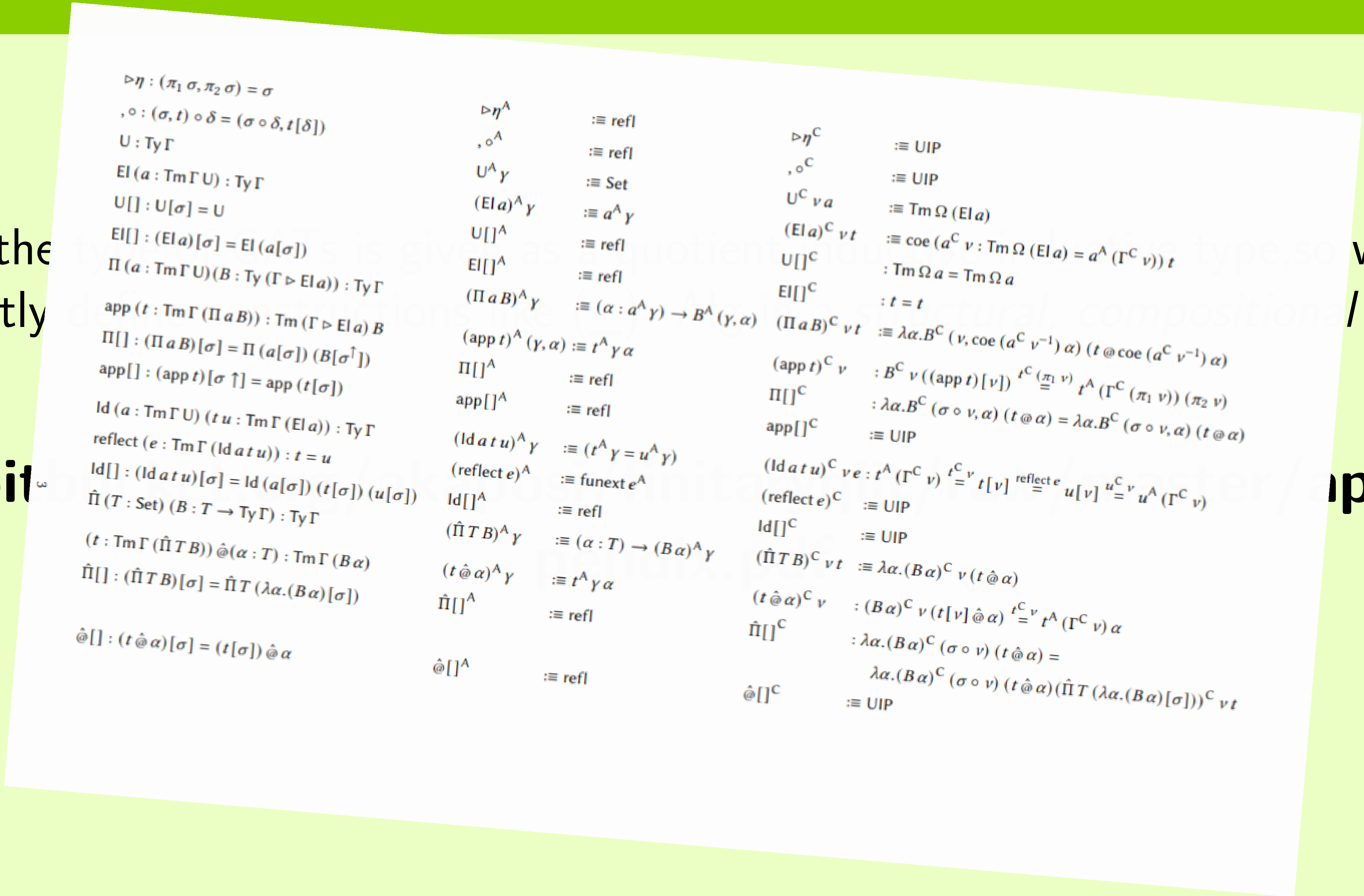
$\cdot\eta^C :\equiv \mathsf{UIP}$

${\rhd\beta_1}^C :\equiv \mathsf{UIP}$

${\rhd\beta_2}^C :\equiv \mathsf{UIP}$

Now, the … we can explicitly … / way.

bit … ap-



$\triangleright\eta : (\pi_1\,\sigma, \pi_2\,\sigma) = \sigma$

$,\circ : (\sigma, t) \circ \delta = (\sigma \circ \delta, t[\delta])$

$\mathsf{U} : \mathsf{Ty}\,\Gamma$

$\mathsf{El}\,(a : \mathsf{Tm}\,\Gamma\,\mathsf{U}) : \mathsf{Ty}\,\Gamma$

$\mathsf{U}[] : \mathsf{U}[\sigma] = \mathsf{U}$

$\mathsf{El}[] : (\mathsf{El}\,a)[\sigma] = \mathsf{El}\,(a[\sigma])$

$\Pi\,(a : \mathsf{Tm}\,\Gamma\,\mathsf{U})(B : \mathsf{Ty}\,(\Gamma \triangleright \mathsf{El}\,a)) : \mathsf{Ty}\,\Gamma$

$\mathsf{app}\,(t : \mathsf{Tm}\,\Gamma\,(\Pi\,a\,B)) : \mathsf{Tm}\,(\Gamma \triangleright \mathsf{El}\,a)\,B$

$\Pi[] : (\Pi\,a\,B)[\sigma] = \Pi\,(a[\sigma])\,(B[\sigma^{\uparrow}])$

$\mathsf{app}[] : (\mathsf{app}\,t)[\sigma \uparrow] = \mathsf{app}\,(t[\sigma])$

$\mathsf{Id}\,(a : \mathsf{Tm}\,\Gamma\,\mathsf{U})\,(t\,u : \mathsf{Tm}\,\Gamma\,(\mathsf{El}\,a)) : \mathsf{Ty}\,\Gamma$

$\mathsf{reflect}\,(e : \mathsf{Tm}\,\Gamma\,(\mathsf{Id}\,a\,t\,u)) : t = u$

$\mathsf{Id}[] : (\mathsf{Id}\,a\,t\,u)[\sigma] = \mathsf{Id}\,(a[\sigma])\,(t[\sigma])\,(u[\sigma])$

$\hat{\Pi}\,(T : \mathsf{Set})\,(B : T \to \mathsf{Ty}\,\Gamma) : \mathsf{Ty}\,\Gamma$

$(t : \mathsf{Tm}\,\Gamma\,(\hat{\Pi}\,T\,B))\,\hat{@}(\alpha : T) : \mathsf{Tm}\,\Gamma\,(B\,\alpha)$

$\hat{\Pi}[] : (\hat{\Pi}\,T\,B)[\sigma] = \hat{\Pi}\,T\,(\lambda\alpha.(B\,\alpha)[\sigma])$

$\hat{@}[] : (t\,\hat{@}\,\alpha)[\sigma] = (t[\sigma])\,\hat{@}\,\alpha$

$\triangleright\eta^{\mathsf{A}} :\equiv \mathsf{refl}$

$,\circ^{\mathsf{A}} :\equiv \mathsf{refl}$

$\mathsf{U}^{\mathsf{A}}\,\gamma :\equiv \mathsf{Set}$

$(\mathsf{El}\,a)^{\mathsf{A}}\,\gamma :\equiv a^{\mathsf{A}}\,\gamma$

$\mathsf{U}[]^{\mathsf{A}} :\equiv \mathsf{refl}$

$\mathsf{El}[]^{\mathsf{A}} :\equiv \mathsf{refl}$

$(\Pi\,a\,B)^{\mathsf{A}}\,\gamma :\equiv (\alpha : a^{\mathsf{A}}\,\gamma) \to B^{\mathsf{A}}\,(\gamma, \alpha)$

$(\mathsf{app}\,t)^{\mathsf{A}}\,(\gamma, \alpha) :\equiv t^{\mathsf{A}}\,\gamma\,\alpha$

$\Pi[]^{\mathsf{A}} :\equiv \mathsf{refl}$

$\mathsf{app}[]^{\mathsf{A}} :\equiv \mathsf{refl}$

$(\mathsf{Id}\,a\,t\,u)^{\mathsf{A}}\,\gamma :\equiv (t^{\mathsf{A}}\,\gamma = u^{\mathsf{A}}\,\gamma)$

$(\mathsf{reflect}\,e)^{\mathsf{A}} :\equiv \mathsf{funext}\,e^{\mathsf{A}}$

$\mathsf{Id}[]^{\mathsf{A}} :\equiv \mathsf{refl}$

$(\hat{\Pi}\,T\,B)^{\mathsf{A}}\,\gamma :\equiv (\alpha : T) \to (B\,\alpha)^{\mathsf{A}}\,\gamma$

$(t\,\hat{@}\,\alpha)^{\mathsf{A}}\,\gamma :\equiv t^{\mathsf{A}}\,\gamma\,\alpha$

$\hat{\Pi}[]^{\mathsf{A}} :\equiv \mathsf{refl}$

$\hat{@}[]^{\mathsf{A}} :\equiv \mathsf{refl}$

$\triangleright\eta^{\mathsf{C}} :\equiv \mathsf{UIP}$

$,\circ^{\mathsf{C}} :\equiv \mathsf{UIP}$

$\mathsf{U}^{\mathsf{C}}\,\nu\,a :\equiv \mathsf{Tm}\,\Omega\,(\mathsf{El}\,a)$

$(\mathsf{El}\,a)^{\mathsf{C}}\,\nu\,t :\equiv \mathsf{coe}\,(a^{\mathsf{C}}\,\nu : \mathsf{Tm}\,\Omega\,(\mathsf{El}\,a) = a^{\mathsf{A}}\,(\Gamma^{\mathsf{C}}\,\nu))\,t$

$\mathsf{U}[]^{\mathsf{C}} : \mathsf{Tm}\,\Omega\,a = \mathsf{Tm}\,\Omega\,a$

$\mathsf{El}[]^{\mathsf{C}} : t = t$

$(\Pi\,a\,B)^{\mathsf{C}}\,\nu\,t :\equiv \lambda\alpha.B^{\mathsf{C}}\,(\nu, \mathsf{coe}\,(a^{\mathsf{C}}\,\nu^{-1})\,\alpha)\,(t\,@\,\mathsf{coe}\,(a^{\mathsf{C}}\,\nu^{-1})\,\alpha)$

$(\mathsf{app}\,t)^{\mathsf{C}}\,\nu : B^{\mathsf{C}}\,\nu\,((\mathsf{app}\,t)[\nu]) \overset{t^{\mathsf{C}}\,(\pi_1\,\nu)}{=} t^{\mathsf{A}}\,(\Gamma^{\mathsf{C}}\,(\pi_1\,\nu))\,(\pi_2\,\nu)$

$\Pi[]^{\mathsf{C}} : \lambda\alpha.B^{\mathsf{C}}\,(\sigma \circ \nu, \alpha)\,(t\,@\,\alpha) = \lambda\alpha.B^{\mathsf{C}}\,(\sigma \circ \nu, \alpha)\,(t\,@\,\alpha)$

$\mathsf{app}[]^{\mathsf{C}} :\equiv \mathsf{UIP}$

$(\mathsf{Id}\,a\,t\,u)^{\mathsf{C}}\,\nu\,e : t^{\mathsf{A}}\,(\Gamma^{\mathsf{C}}\,\nu) \overset{t^{\mathsf{C}}\,\nu}{=} t[\nu] \overset{\mathsf{reflect}\,e}{=} u[\nu] \overset{u^{\mathsf{C}}\,\nu}{=} u^{\mathsf{A}}\,(\Gamma^{\mathsf{C}}\,\nu)$

$(\mathsf{reflect}\,e)^{\mathsf{C}} :\equiv \mathsf{UIP}$

$\mathsf{Id}[]^{\mathsf{C}} :\equiv \mathsf{UIP}$

$(\hat{\Pi}\,T\,B)^{\mathsf{C}}\,\nu\,t :\equiv \lambda\alpha.(B\,\alpha)^{\mathsf{C}}\,\nu\,(t\,\hat{@}\,\alpha)$

$(t\,\hat{@}\,\alpha)^{\mathsf{C}}\,\nu : (B\,\alpha)^{\mathsf{C}}\,\nu\,(t[\nu]\,\hat{@}\,\alpha) \overset{t^{\mathsf{C}}\,\nu}{=} t^{\mathsf{A}}\,(\Gamma^{\mathsf{C}}\,\nu)\,\alpha$

$\hat{\Pi}[]^{\mathsf{C}} : \lambda\alpha.(B\,\alpha)^{\mathsf{C}}\,(\sigma \circ \nu)\,(t\,\hat{@}\,\alpha) = \lambda\alpha.(B\,\alpha)^{\mathsf{C}}\,(\sigma \circ \nu)\,(t\,\hat{@}\,\alpha)(\hat{\Pi}\,T\,(\lambda\alpha.(B\,\alpha)[\sigma]))^{\mathsf{C}}\,\nu\,t$

$\hat{@}[]^{\mathsf{C}} :\equiv \mathsf{UIP}$

2 Concrete CwFs

Central Dogma of Category Theory

Every notion of “structure” comes equipped with a notion of “structure-preserving morphism”

Central Dogma of Generalized Algebra

Every notion of “structure” comes equipped with a notion of “structure-preserving morphism”, “displayed structure”, and “section”

Displayed Nat-Algebra is induction data

$$
\begin{aligned}
\mathfrak{N}\text{ - DAlg }(N, z, s) = {} \\
&(N^{\mathrm{D}} \colon N \to \mathrm{Set}) \\
\times\ &(z^{\mathrm{D}} \colon N^{\mathrm{D}}(z)) \\
\times\ &(s^{\mathrm{D}} \colon (n \colon N) \to N^{\mathrm{D}}(n) \to N^{\mathrm{D}}(s\ n)) \\
\mathfrak{N}\text{ - Sect }(N, z, s)\ (N^{\mathrm{D}}, z^{\mathrm{D}}, s^{\mathrm{D}}) = {} \\
&(N^{\mathrm{S}} \colon (n \colon N) \to N^{\mathrm{D}}(n)) \\
\times\ &(N^{\mathrm{S}}(z) = z^{\mathrm{D}}) \\
\times\ &((n \colon N) \to N^{\mathrm{S}}(s\ n) = s^{\mathrm{D}}\ n\ (N^{\mathrm{S}}\ n))
\end{aligned}
$$

Every displayed algebra over the initial algebra admits a section

- **Induction**: From a predicate with sufficient data, obtain a section by induction
- **Unary Parametricity**

$$\{\!\![\mathtt{X} : \mathbf{U}, \mathtt{x} : \mathtt{X}]\!\!\} \vdash \mathtt{t} : \mathtt{X}$$

Proof irrelevant displayed algebras

$$
\begin{aligned}
\mathfrak{Grp}\text{-}\mathsf{DAlg}\,(M, u, \mu, \_, \_, \_, i, \_, \_) = {}& \\
& (M^{\mathsf{D}} \colon M \to \mathsf{Prop}) \\
\times\ & (u^{\mathsf{D}} \colon M^{\mathsf{D}}(u)) \\
\times\ & (\mu^{\mathsf{D}} \colon (m_0\ m_1 \colon M) \to M^{\mathsf{D}}(m_0) \to M^{\mathsf{D}}(m_1) \to M^{\mathsf{D}}(\mu(m_0, m_1))) \\
\times\ & \ldots \\
\times\ & \ldots \\
\times\ & \ldots \\
\times\ & ((m \colon M) \to M^{\mathsf{D}}(m) \to M^{\mathsf{D}}(i\ m)) \\
\times\ & \ldots \\
\times\ & \ldots
\end{aligned}
$$

Notice something...

$$
\begin{aligned}
\mathfrak{G}\text{-}\mathsf{Alg} &: \mathsf{Set} \\
\_ \to \_ &: \mathfrak{G}\text{-}\mathsf{Alg} \to \mathfrak{G}\text{-}\mathsf{Alg} \to \mathsf{Set} \\
\mathfrak{G}\text{-}\mathsf{DAlg} &: \mathfrak{G}\text{-}\mathsf{Alg} \to \mathsf{Set} \\
\mathfrak{G}\text{-}\mathsf{Sect} &: (\Gamma : \mathfrak{G}\text{-}\mathsf{Alg}) \to \mathfrak{G}\text{-}\mathsf{DAlg} \to \mathsf{Set}
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{Con} &: \mathsf{Set} \\
\mathsf{Sub} &: \mathsf{Con} \to \mathsf{Con} \to \mathsf{Set} \\
\mathsf{Ty} &: \mathsf{Con} \to \mathsf{Set} \\
\mathsf{Tm} &: (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\ \Gamma \to \mathsf{Set}
\end{aligned}
$$

The Algebras of a GAT form a CwF!

Central Dogma of Category Theory

Every notion of "structure" comes equipped with a notion of "structure-preserving morphism", i.e. ***forms a category***

Central Dogma of Generalized Algebra

Every notion of "structure" comes equipped with a notion of "structure-preserving morphism", "displayed structure", and "section", i.e. ***forms a category with families***

“Concrete CwFs”

The CwF of CwFs?

3 Fibrancy and Autosynthesis

Is the concrete CwF of setoids the same thing as the ***setoid model***?

Is the concrete CwF of groupoids the same thing as the ***groupoid model***?

No

Question: Can we do this more generally?

Fibrancy Question

For which GATs can we articulate an appropriate notion of “fibrancy” for their concrete CwF’s types (i.e. their displayed algebras)?

Cosmic Question

For which GATs $\mathfrak{G}$ can the category of $\mathfrak{G}$-algebras be viewed as a $\mathfrak{G}$-algebra?

Autosynthesis Question

For which GATs $\mathfrak{G}$ does (some fibrant version of) their concrete CwF interpret a synthetic theory of $\mathfrak{G}$-algebras?

Work on these questions for category-like GATs

- **Setoids**: ✓ (Hofmann, Altenkirch,...)
- **Groupoids**: ✓ (Hofmann & Streicher)
- **Categories**:
  - Cosmic ✓ (Lawvere,...)
  - Fibrancy ✓ (Grothendieck,...)
  - Autosynthesis Most directedTT/synthetic CT is in a different direction; work remains to be done (Harper & Licata, North,...)

My PhD thesis: Work the category case out fully

Thanks for listening!

github.com/jacobneu/GeneralizedAlgebra

bitbucket.org/akaposi/finitaryqiit/raw/master/ap-
pendix.pdf