# For Generalised Algebraic Theories, Two Sorts Are Enough

Samy Avrillon, Ambrus Kaposi, Ambroise Lafont, **Niyousha Najmaei** and Johann Rosain

TYPES, Gothenburg
8 May, 2026

Preprint: `https://arxiv.org/abs/2601.19426`

# For Generalised Algebraic Theories, Two Sorts Are Enough

**and by sorts I mean "family sorts"**

Samy Avrillon, Ambrus Kaposi, Ambroise Lafont, **Niyousha Najmaei** and Johann Rosain

TYPES, Gothenburg
8 May, 2026

Preprint: `https://arxiv.org/abs/2601.19426`

# Problem: I Want To Use (Q)IITs

**Rocq:**

- Does not support IITs natively[1].

**Cubical Agda:**

- Supports a wider class of inductive types natively.
- Okay! I will use Cubical Agda.

[1] IITs can be reduced to indexed inductive types Kaposi, Kovács, and Lafont 2019; Sestini 2023, but in extensional/observational type theory

# The QIIT Specification of MLTT

```
Con : Set
Sub : Con → Con → Set
Ty  : Con → Set
Tm  : (Γ : Con) → Ty Γ → Set
...
-- constructors:
_[_] : Ty Γ → Sub Δ Γ → Ty Δ
id   : Sub Γ Γ
...
-- path constructor:
[id] : A [ id ] = A
```

- Multiple inductive types being defined together
- The inductive types are indexed over each other
- Path constructors

# A Family Encoding Is Used

Altenkirch, Kaposi, and Xie 2025 use a "family" encoding.

[1] Altenkirch and Scoccola 2020 also use such a family encoding for defining the HIT of integers.

# A Family Encoding Is Used

Altenkirch, Kaposi, and Xie 2025 use a "family" encoding.

```
Con : Set
Sub : Con → Con → Set
Ty  : Con → Set
Tm  : (Γ : Con) → Ty Γ → Set
...
_[_] : Ty Γ → Sub Δ Γ → Ty Δ
...
```

[1] Altenkirch and Scoccola 2020 also use such a family encoding for defining the HIT of integers.

# A Family Encoding Is Used

Altenkirch, Kaposi, and Xie 2025 use a "family" encoding.

1. Add sorts $U : \mathrm{Set}$ and $\mathrm{El} : U \to \mathrm{Set}$

$U : \mathrm{Set}$

$\mathrm{El} : U \longrightarrow \mathrm{Set}$

```
Con : Set
Sub : Con → Con → Set
Ty  : Con → Set
Tm  : (Γ : Con) → Ty Γ → Set
...
_[_] : Ty Γ → Sub Δ Γ → Ty Δ
...
```

[1] Altenkirch and Scoccola 2020 also use such a family encoding for defining the HIT of integers.

# A Family Encoding Is Used

Altenkirch, Kaposi, and Xie 2025 use a "family" encoding.

1. Add sorts $U : \mathrm{Set}$ and $\mathrm{El} : U \to \mathrm{Set}$
2. Replace Set in the original sorts with $U$

```
U : Set
El  : U → Set
Con : ~~Set~~ ⋃
Sub : Con → Con → ~~Set~~ ⋃
Ty  : Con → ~~Set~~ ⋃
Tm  : (Γ : Con) → Ty Γ → ~~Set~~ ⋃
...
_[_] : Ty Γ → Sub Δ Γ → Ty Δ
...
```

[1] Altenkirch and Scoccola 2020 also use such a family encoding for defining the HIT of integers.

# A Family Encoding Is Used

Altenkirch, Kaposi, and Xie 2025 use a "family" encoding.

1. Add sorts $U : \mathrm{Set}$ and $\mathrm{El} : U \to \mathrm{Set}$
2. Replace Set in the original sorts with $U$
3. Insert El in front of all the original sorts

```
U : Set
El  : U → Set
Con : U
Sub : El Con → El Con → U
Ty  : El Con → U
Tm  : (Γ : El Con) → El Ty Γ → U
...
_[_] : El Ty Γ → El Sub Δ Γ → El Ty Δ
...
```

[1] Altenkirch and Scoccola 2020 also use such a family encoding for defining the HIT of integers.

# A Family Encoding Is Used

Altenkirch, Kaposi, and Xie 2025 use a "family" encoding.

1. Add sorts $U : \mathrm{Set}$ and $\mathrm{El} : U \to \mathrm{Set}$
2. Replace Set in the original sorts with $U$
3. Insert El in front of all the original sorts

```
U : Set
El  : U → Set
Con : U
Sub : El Con → El Con → U
Ty  : El Con → U
Tm  : (Γ : El Con) → El (Ty Γ) → U
...
_[_] : El Ty Γ → El Sub Δ Γ → El Ty Δ
...
```

[1] Altenkirch and Scoccola 2020 also use such a family encoding for defining the HIT of integers.

# Why Use The Family Encoding

**Rocq:**

- Does not support IITs natively.

**Cubical Agda:**

- Supports a wider class of inductive types natively.
- Does **not** support **sort equations** in QIITs.
- Does **not** support **interleaved constructors of different sorts**.
- Does **not** support **interleaved sorts and constructors**.

The family encoding bypasses these restrictions in Cubical Agda.

# Why Use These Encodings: Interleaved constructors

Cubical Agda does **not** support interleaving constructors of different sorts.

```
...
-- some constructors:
_[_] : Ty Γ → Sub Δ Γ → Ty Δ
_∘_ : Sub Δ Γ → Sub Θ Δ → Sub Θ Γ
_▷_ : (Γ : Con) → Ty Γ → Con
...
-- this equality of sort Ty uses the _∘_ constructor of sort Sub:
[∘] : A [ γ ∘ δ ] = A [ γ ] [ δ ]
-- this constructor of sort Sub, uses the _[_] constructor of sort Ty:
_,_ : (g : Sub Δ Γ) → Tm Δ (A [ g ]) → Sub Δ (Γ ▷ A)
...
```

# Why Use These Encodings: Interleaved constructors

After the family encoding:

```
U  : Set
El : U → Set
...
-- constructors now all target El, no interleaving:
_[_] : El (Ty Γ) → El (Sub Δ Γ) → El (Ty Δ)
_∘_ : El (Sub Δ Γ) → El (Sub Θ Δ) → El (Sub Θ Γ)
_▷_ : (Γ : El Con) → El (Ty Γ) → El Con
...
[∘] : A [ γ ∘ δ ] = A [ γ ] [ δ ]
_,_ : (g : El (Sub Δ Γ)) → El (Tm Δ (A [ g ])) → El (Sub Δ (Γ ▷ A))
...
```

# Why Use These Encodings: Sort Equations

**Example:** Extending MLTT with a Russell universe; adding a type of types.

$$\frac{\Gamma \text{ ctx}}{\Gamma \vdash \mathsf{Univ}_\Gamma \text{ type}} \qquad \frac{\Gamma \vdash A : \mathsf{Univ}_\Gamma}{\Gamma \vdash A \text{ type}}$$

# Why Use These Encodings: Sort Equations

**Example:** Extending MLTT with a Russell universe; adding a type of types.

$$\frac{\Gamma \text{ ctx}}{\Gamma \vdash \mathsf{Univ}_\Gamma \text{ type}} \qquad \frac{\Gamma \vdash A : \mathsf{Univ}_\Gamma}{\Gamma \vdash A \text{ type}}$$

For our specification: Adding a constructor and a **sort equation** expressing that terms of type $\mathsf{Univ}\,\Gamma$ are identified by types in $\Gamma$.

$$\mathsf{Univ} : (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \qquad \text{and} \qquad \mathsf{Tm}\,\Gamma\,(\mathsf{Univ}\,\Gamma) =_{\mathsf{Set}} \mathsf{Ty}\,\Gamma$$

Sort equations are **not** accepted by Cubical Agda.

# Why Use These Encodings: Sort Equations

After the family encoding:

$$\mathsf{Tm}\,\Gamma\,(\mathsf{Univ}\,\Gamma) =_{\mathsf{Set}} \mathsf{Ty}\,\Gamma \qquad \text{becomes} \qquad \mathsf{Tm}\,\Gamma\,(\mathsf{Univ}\,\Gamma) =_{U} \mathsf{Ty}\,\Gamma$$

```
U    : Set
El   : U → Set
Con  : U
Sub  : El Con → El Con → U
Ty   : El Con → U
Tm   : (Γ : El Con) → El (Ty Γ) → U
...
Univ : (Γ : El Con) → El (Ty Γ)
-- sort equation becomes equality in U
UEq  : Tm Γ (Univ Γ) = Ty Γ
...
```

# Does It Always Work?

- Sestini 2023 defined a reduction of family inductive-inductive types to W-types in observational type theory, and **conjectured the existence of a general family encoding** to justify focusing on the family IITs.

# Does It Always Work?

**What do we mean by "it works"?**

We want a construction that takes the encoded (Q)IIT and:

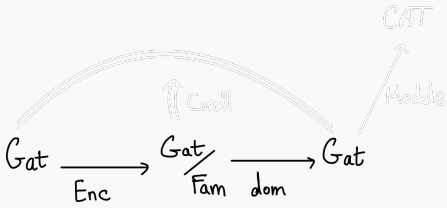
- gives a model of the original QIIT
- equips this model with a suitable recursion principle (aka initiality principle)

In categorical terms: a functor $\mathrm{Models}(\mathrm{Enc}(\Gamma)) \to \mathrm{Models}(\Gamma)$ that preserves initial objects.

# It Always Works!

Generalise from (Q)IIT specifications to arbitrary generalised algebraic theories (GATs):

1. family encoding[1]: Enc; fully faithful

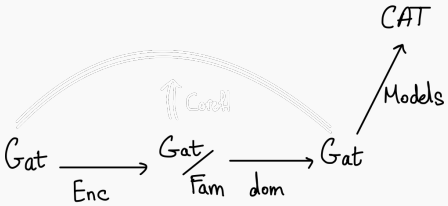



[1]Direct use of Uemura's characterisation of GATs

# It Always Works!

Generalise from (Q)IIT specifications to arbitrary generalised algebraic theories (GATs):

1. family encoding[1]: Enc; fully faithful
2. category of models[1]: Models

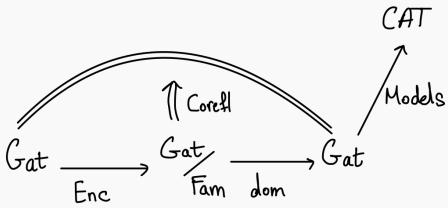



[1]Direct use of Uemura's characterisation of GATs

# It Always Works!

Generalise from (Q)IIT specifications to arbitrary generalised algebraic theories (GATs):

1. family encoding[1]: Enc; fully faithful
2. category of models[1]: Models
3. morphisms $\mathrm{Enc}(\Gamma) \to \Gamma$

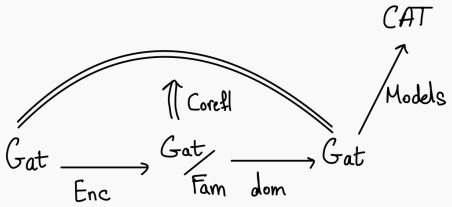



[1]Direct use of Uemura's characterisation of GATs

# It Always Works!

Generalise from (Q)IIT specifications to arbitrary generalised algebraic theories (GATs):

1. family encoding[1]: Enc; fully faithful
2. category of models[1]: Models
3. morphisms $\mathrm{Enc}(\Gamma) \to \Gamma$
4. a **strict coreflection**: adjunction with $\eta = \mathrm{Id}$

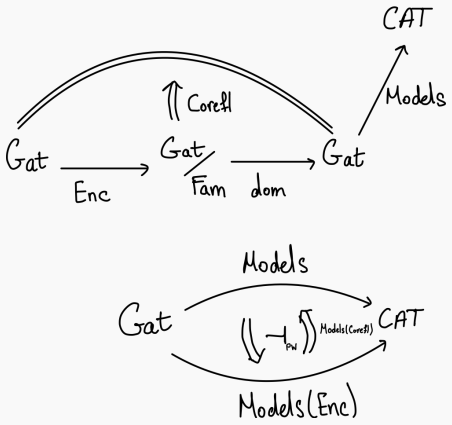



[1]Direct use of Uemura's characterisation of GATs

# It Always Works!

Generalise from (Q)IIT specifications to arbitrary generalised algebraic theories (GATs):

1. family encoding[1]: Enc; fully faithful
2. category of models[1]: Models
3. morphisms $\mathrm{Enc}(\Gamma) \to \Gamma$
4. a **strict coreflection**: adjunction with $\eta = \mathrm{Id}$
   $\Rightarrow$ the **right** adjoint applied to the initial model of $\mathrm{Enc}(\Gamma)$ is the initial model of $\Gamma$




[1]Direct use of Uemura's characterisation of GATs

# The Coreflection

We want

$$\mathsf{Models}(\Gamma) \underset{\mathsf{Models}(\mathsf{Corefl}_\Gamma)}{\overset{\longrightarrow}{\underset{\longleftarrow}{\bot}}} \mathsf{Models}(\mathsf{Enc}(\Gamma))$$

1. characterise $\mathsf{Models}(\mathsf{Enc}(\Gamma))$ as being iso to a kind of comma category $F_\Gamma/_c\mathsf{Fam}$
2. the strict coreflection becomes easy to check

# Models of the Reduced GAT

We define the family functor $F_\Gamma : \mathsf{Models}(\Gamma) \to \mathsf{Fam}$[1].

**Theorem**

*Given* $\Gamma$ *with family functor* $F_\Gamma : \mathsf{Models}(\Gamma) \to \mathsf{Fam}$, *we have* $\mathsf{Models}(\mathsf{Enc}(\Gamma))$ *is iso to the category whose objects are "cartesian" morphisms of families of the form*

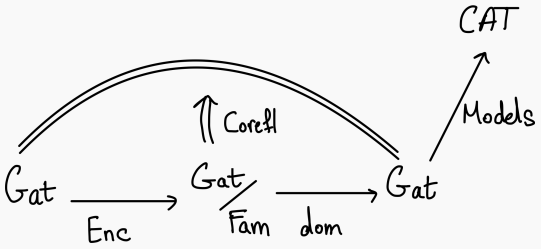
$$F_\Gamma(M) \to (U', El')$$

*for some model* $M \in \mathsf{Models}(\Gamma)$ *and family* $(U', El')$.

[1] Using Uemura's characterisation of GATs

# Universal Property of (Finite) GATs

# Back to The Other Components




Hammer: Uemura's characterisation of GATs

# The Universal Property of GAT

Informally, FinGat is the category of finite GATs and GAT morphisms (substitutions).

**Theorem (Uemura 2022)**

**FinGat**, *equipped with* $p_G : (A : \mathrm{Set},\ a : A) \to (A : \mathrm{Set})$, *is* ***bi-initial*** *among categories with* ***finite limits*** *and a choice of an "exponentiable morphism"*[2].

[2] exponentiable object in a slice

# The Universal Property of GAT

Informally, FinGat is the category of finite GATs and GAT morphisms (substitutions).

**Theorem (Uemura 2022)**

**FinGat**, *equipped with* $p_G : (A : \mathrm{Set},\ a : A) \to (A : \mathrm{Set})$, *is* ***bi-initial*** *among categories with* ***finite limits*** *and a choice of an "exponentiable morphism"*[2].

**Theorem**

**Gat**, *equipped with* $p_G : (A : \mathrm{Set},\ a : A) \to (A : \mathrm{Set})$, *is* ***bi-initial*** *among categories with* ***limits*** *and a choice of an "exponentiable morphism".*

[2] exponentiable object in a slice

# Uemura's Characterisation — Consequence

**Key consequence:** to define a functor out of FinGat it suffices to:

- show that the target category has finite limits
- check that the morphism we want $p_G$ to be mapped to is exponentiable

Then, define the functor using bi-initiality

**Example (The Model Functor)**

Let's define a functor

$$\text{Models} \ : \ \text{FinGat} \longrightarrow \text{CAT}$$

mapping each theory $\Gamma$ to its **category of models** Models($\Gamma$).

We want $p_G : (A : \text{Set},\ a : A) \to (A : \text{Set})$ to be mapped to PtdSet $\to$ Set. So we check that PtdSet $\to$ Set is exponentiable.

Bi-initiality then yields a functor:

$$\text{Models} \ : \ \text{FinGat} \longrightarrow \text{CAT}$$

# Family Encoding

**Example (Family Encoding)**

Let's define a functor

$$\mathrm{Enc} \;:\; \mathrm{Gat} \longrightarrow \mathrm{Gat/Fam}$$

mapping $\Gamma$ to $\mathrm{Enc}(\Gamma)$, **the family encoded GAT** with sorts $(U : \mathrm{Set},\ El : U \to \mathrm{Set})$.

We want $p_G : (A : \mathrm{Set},\ a : A) \to (A : \mathrm{Set})$ to be mapped to

$$(U : \mathrm{Set},\ El : U \to \mathrm{Set},\ A : U,\ a : El\, A) \longrightarrow (U : \mathrm{Set},\ El : U \to \mathrm{Set},\ A : U).$$

We show that this is exponentiable using standard results about exponentiable morphisms. Bi-initiality yields Enc.

# Intuition

For more intuition about Uemura's characterisation, for example the relation between context extension and $p_G$: see Uemura 2022 and our TYPES abstract.

## Contributions

1. Adapting Uemura's universal property for FinGat to Gat
   - Our main technical tool
2. Family encoding functor Enc : Gat $\to$ Gat/Fam, mapping any GAT to a family GAT with sorts $U$ : Set and $El : U \to$ Set
3. Characterisation of Models(Enc(Γ)) as a kind of comma category "$F_\Gamma/_c Fam$"
4. Consequence of point 3: a strict coreflection Models(Γ) $\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}$ Models(Enc(Γ))
5. Proving the family encoding functor is fully faithful
   - For this we show the existence of initial models

# Thank you!

# References i



Altenkirch, Thorsten, Ambrus Kaposi, and Szumi Xie (2025). **"The Groupoid-syntax of Type Theory is a Set".** In: *CoRR* abs/2509.14988. DOI: 10.48550/ARXIV.2509.14988. arXiv: 2509.14988. URL: https://doi.org/10.48550/arXiv.2509.14988.


Altenkirch, Thorsten and Luis Scoccola (2020). **"The Integers as a Higher Inductive Type".** In: *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*. Ed. by Holger Hermanns et al. ACM, pp. 67–73. DOI: 10.1145/3373718.3394760. URL: https://doi.org/10.1145/3373718.3394760.

# References ii


- Kaposi, Ambrus, András Kovács, and Ambroise Lafont (2019). **"For Finitary Induction-Induction, Induction Is Enough".** In: *25th International Conference on Types for Proofs and Programs, TYPES 2019, Oslo, Norway, June 11-14, 2019*. Ed. by Marc Bezem and Assia Mahboubi. Vol. 175. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 6:1–6:30. DOI: 10.4230/LIPICS.TYPES.2019.6. URL: https://doi.org/10.4230/LIPIcs.TYPES.2019.6.
- Sestini, Filippo (Dec. 2023). **"Bootstrapping extensionality".** PhD thesis. URL: https://eprints.nottingham.ac.uk/74363/.
- Uemura, Taichi (2022). **"The universal exponentiable arrow".** In: *Journal of Pure and Applied Algebra* 226.7, p. 106991. ISSN: 0022-4049. DOI: https://doi.org/10.1016/j.jpaa.2021.106991. URL: https://www.sciencedirect.com/science/article/pii/S0022404921003327.

# Uemura's Characterisation

**Definition**

A **CartExp-category** is a category with finite limits equipped with a chosen *exponentiable* morphism $p : Y \to X$ (i.e. the pullback functor $p^* : \mathcal{C}/X \to \mathcal{C}/Y$ has a right adjoint).

A **CartExp-functor** preserves finite limits, the chosen morphism, and the right adjoint.

**Informally:** FinGat has finite GATs as objects and substitutions as morphisms.

**Theorem (Uemura 2022)**

**FinGat**, *equipped with* $p_G : (A : \mathrm{Set},\ a : A) \to (A : \mathrm{Set})$, *is* ***bi-initial*** *in the 2-category of CartExp-categories: for any CartExp-category* $(C, p')$ *there exists a CartExp-functor* FinGat $\to C$ *sending* $p_G$ *to* $p'$, *unique up to unique isomorphism.*

# Models of the Reduced GAT: Key Ingredient for Coreflection

We define the family functor $F_\Gamma : \mathrm{Models}(\Gamma) \to \mathrm{Fam}$ using bi-initiality of Gat.

**Theorem**

*Given* $\Gamma$ *with family functor* $F_\Gamma : \mathrm{Models}(\Gamma) \to \mathrm{Fam}$, *we have* $\mathrm{Models}(\mathrm{Enc}(\Gamma))$ *is iso to the category whose objects are "cartesian" morphisms of families of the form*

$$F_\Gamma(M) \to (U', El')$$

*for some model* $M \in \mathrm{Models}(\Gamma)$ *and family* $(U', El')$.

# The Coreflection

Now that we have $\mathsf{Models}(\mathsf{Enc}(\Gamma)) \cong F_\Gamma/_c\mathsf{Fam}$, we can define the coreflection.

$$\mathsf{Models}(\Gamma) \underset{\mathsf{Models}(\mathsf{Coref}\mathsf{l}_\Gamma)}{\overset{\longrightarrow}{\underset{\longleftarrow}{\bot}}} \mathsf{Models}(\mathsf{Enc}(\Gamma))$$

- The **right adjoint** $\mathsf{Models}(\mathsf{Coref}\mathsf{l}_\Gamma)$ maps a cartesian morphism $F_\Gamma(M) \to (U', El')$ to its underlying model $M$.
- The **left adjoint** maps a model $M$ to the identity $F_\Gamma(M) \xrightarrow{\mathrm{id}} F_\Gamma(M)$.
- The unit is the **identity**: *strict* coreflection.
- Round-trip starting from any model of $\Gamma$ is the identity.

# Example of the Family Encoding Functor and Coreflector

**Example (**$\mathrm{Enc}(A : \mathrm{Set})$**)**

The functor Enc maps the theory $\Gamma := (A : \mathrm{Set})$ to:

$$U : \mathrm{Set}, \quad El : U \to \mathrm{Set}, \quad A : U.$$

Its coreflector morphism is the following morphism:

$$\mathrm{Corefl}_\Gamma : (U : \mathrm{Set},\ El : U \to \mathrm{Set},\ A : U) \longrightarrow (A : \mathrm{Set})$$

which interprets the sort $A : \mathrm{Set}$ of $\Gamma$ as the term $El\, A : \mathrm{Set}$ in $\mathrm{Enc}(\Gamma)$.

# Example: Transitive Graphs

The GAT of transitive graphs:

```
V : Set
E : V → V → Set
T : (v1 v2 v3 : V) → E v1 v2 → E v2 v3 → E v1 v3
```

# The Family Functor: A Key Ingredient

Using bi-initiality of Gat, we get the family functor $F_\Gamma : \mathrm{Models}(\Gamma) \to \mathrm{Fam}$.

**Example ($A : \mathrm{Set}$)**

The family functor of $(A : \mathrm{Set})$ maps a set $X$ to the family $(U, El)$ defined by $U = \{*\}$ and $El(*) = X$.

**Example (Transitive Graphs)**

For transitive graphs, $F_\Gamma$ maps a model $(V, E, T)$ to the family $(U_G, El_G)$ defined by $U_G = \{*\} + V \times V$, $El_G(*) = V$, and $El_G(a, b) = E(a, b)$.

# Example of the Family Encoding Functor and Coreflector

**Example (Enc(Transitive Graphs))**

The functor Enc maps the GAT of transitive graphs to:

$$U : \mathrm{Set}, \quad El : U \to \mathrm{Set}, \quad V : U, \quad E : El\ V \to El\ V \to U$$

$$T : \prod_{v_1, v_2, v_3 : El\ V} El(E\ v_1\ v_2) \to El(E\ v_2\ v_3) \to El(E\ v_1\ v_3)$$

The coreflector morphism is the GAT morphism $\mathrm{Corefl}_\Gamma : \mathrm{Enc}(\Gamma) \longrightarrow \Gamma$ which interprets:

- the sort $V : \mathrm{Set}$ of $\Gamma$ as the term $El\ V : \mathrm{Set}$ in $\mathrm{Enc}(\Gamma)$,
- the sort $E : V \to V \to \mathrm{Set}$ of $\Gamma$ as the term $\lambda\, v_1\ v_2.\ El(E\ v_1\ v_2) : \mathrm{Set}$ in $\mathrm{Enc}(\Gamma)$,
- the operator $T$ of $\Gamma$ as the operator $T$ of $\mathrm{Enc}(\Gamma)$.

# Example of the Characterisation of Models(Enc(Γ))

**Example (What is Models(Enc((A : Set))) via $F_{A:Set}/_c$Fam)**

For $\Gamma := (A : \mathrm{Set})$, an object of $F_\Gamma/_c\mathrm{Fam}$ consists of:

- a set $X$,
- a family $(U', El')$
- a function $f : \{*\} \to U'$ such that $El'(f(*)) = X$.

Giving $f$ is choosing an element $A' \in U'$ with $El'(A') = X$. So $(U', El', A')$ is precisely a model of $T(A : \mathrm{Set}) = (U : \mathrm{Set},\ El : U \to \mathrm{Set},\ A : U)$.

Also note that, the round-trip $\mathrm{Models}(\Gamma) \to \mathrm{Models}(\mathrm{Enc}(\Gamma)) \to \mathrm{Models}(\Gamma)$ starting from $X$ gives back $X$.

## Example of the Characterisation of Models(Enc(Γ))

**Example (Models(Enc((Transitive Graphs))))**

For $\Gamma :=$ Transitive Graphs, an object of $F_\Gamma/_c\mathrm{Fam}$ consists of:

- a transitive graph $G = (V, E, T)$,
- a family $(U', El')$
- a function $f : \{*\} + V \times V \to U'$ such that $El'(f(*)) = V$ and $El'(f(a, b)) = E(a, b)$ for all $a, b \in V$.

Giving $f$ is choosing:oh

- an element $V' \in U'$ with $El'(V') = V$,
- a function $E' : V \times V \to U'$ with $El'(E'(a, b)) = E(a, b)$.

So $(U', El', V', E', T)$ is precisely a model of $\mathrm{Enc}(\Gamma)$. Also note that, the round-trip $\mathrm{Models}(\Gamma) \to \mathrm{Models}(\mathrm{Enc}(\Gamma)) \to \mathrm{Models}(\Gamma)$ starting from $G$ gives back $G$.