

For Generalised Algebraic Theories, Two Sorts Are Enough

and by two sorts I mean “family sorts”

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

LHC Days, 17 June 2026

Preprint: <https://arxiv.org/abs/2601.19426>

Example: The GAT of MLTT

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

- Multi-sorted
- Sorts indexed over each other
- Equations

The Family Encoding

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
...
```

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

The Family Encoding

Altenkirch, Kaposi, and Xie 2025 use a “family” encoding.

1. Add sorts $U : \text{Set}$ and $\text{El} : U \rightarrow \text{Set}$

```
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
...
```

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

The Family Encoding

Altenkirch, Kaposi, and Xie 2025 use a “family” encoding.

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

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

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

The Family Encoding

Altenkirch, Kaposi, and Xie 2025 use a “family” encoding.

1. Add sorts $U : \text{Set}$ and $\text{El} : U \rightarrow \text{Set}$
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 :  $\checkmark_{\text{El}}$  Con →  $\checkmark_{\text{El}}$  Con → U
Ty  :  $\checkmark_{\text{El}}$  Con → U
Tm  : ( $\Gamma : \checkmark_{\text{El}}$  Con) →  $\checkmark_{\text{El}}$  Ty  $\Gamma$  → U
...
_[] :  $\checkmark_{\text{El}}$  Ty  $\Gamma$  →  $\checkmark_{\text{El}}$  Sub  $\Delta$   $\Gamma$  →  $\checkmark_{\text{El}}$  Ty  $\Delta$ 
...
```

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

The Family Encoding

Altenkirch, Kaposi, and Xie 2025 use a “family” encoding.

1. Add sorts $U : \text{Set}$ and $\text{El} : U \rightarrow \text{Set}$
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
...
```

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

What Does the Family Encoding Buy Us?

Family encoding is a reduction of GATs to ones with sorts $U : \text{Set}$, $El : U \rightarrow \text{Set}$.

In the resulting GAT:

- We don't have **sort equations**
- We don't have **interleaved constructors of different sorts**
- We don't have **interleaved sorts and constructors**

Why Use These Encodings: Interleaved Constructors

```
...
-- some operatios:
_[] : Ty  $\Gamma$   $\rightarrow$  Sub  $\Delta$   $\Gamma$   $\rightarrow$  Ty  $\Delta$ 
_[] : Sub  $\Delta$   $\Gamma$   $\rightarrow$  Sub  $\Theta$   $\Delta$   $\rightarrow$  Sub  $\Theta$   $\Gamma$ 
_▷_ : ( $\Gamma$  : Con)  $\rightarrow$  Ty  $\Gamma$   $\rightarrow$  Con
...
-- this equality of sort Ty uses the _[] operation of sort Sub:
[o] : A [  $\gamma$   $\circ$   $\delta$  ] =Ty A [  $\gamma$  ] [  $\delta$  ]
-- this operation of sort Sub, uses the _[] operation of sort Ty:
_,_ : (g : Sub  $\Delta$   $\Gamma$ )  $\rightarrow$  Tm  $\Delta$  (A [ g ])  $\rightarrow$  Sub  $\Delta$  ( $\Gamma$  ▷ A)
...
```

Why Use These Encodings: Interleaved Constructors

After the family encoding:

```
U : Set
El : U → Set
...
-- operations 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 ] =Ty 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.

$$\frac{\Gamma \text{ ctx}}{\Gamma \vdash \text{Univ}_\Gamma \text{ type}}$$

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

Why Use These Encodings: Sort Equations

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

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

In the GAT: Adding an operation and a **sort equation** expressing that terms of type $\text{Univ } \Gamma$ are identified by types in Γ :

$$\text{Univ} : (\Gamma : \text{Con}) \rightarrow \text{Ty } \Gamma \quad \text{and} \quad \text{Tm } \Gamma (\text{Univ } \Gamma) =_{\text{Set}} \text{Ty } \Gamma$$

Why Use These Encodings: Sort Equations

After the family encoding:

$\text{Tm } \Gamma (\text{Univ } \Gamma) =_{\text{Set}} \text{Ty } \Gamma$ becomes $\text{Tm } \Gamma (\text{Univ } \Gamma) =_U \text{Ty } \Gamma$

```
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 of U
UEq    : Tm Γ (Univ Γ) =_U Ty Γ
...
```

- **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”?

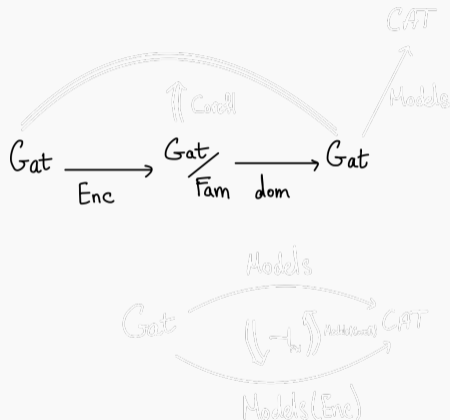
We want a construction that takes the encoded GAT and:

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

In categorical terms: a functor $\text{Models}(\text{Enc}(G)) \rightarrow \text{Models}(G)$ that preserves initial objects.

It Always Works!

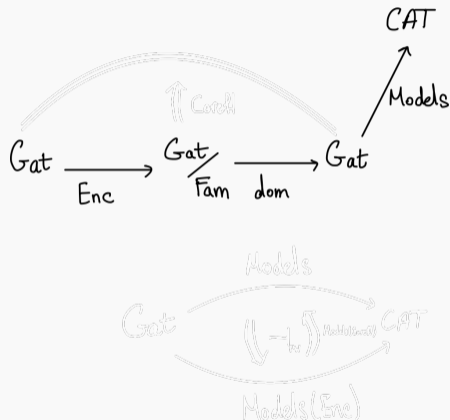
1. family encoding¹: Enc; fully faithful



¹Direct use of Uemura's characterisation of GATs

It Always Works!

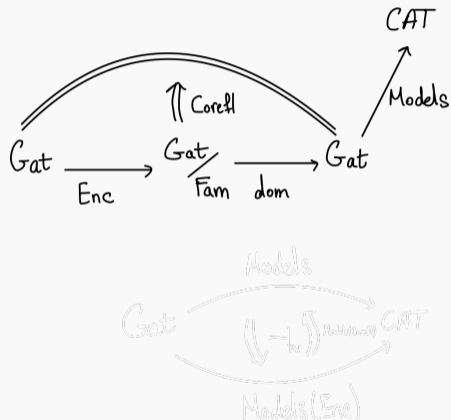
1. family encoding¹: Enc; fully faithful
2. category of models¹: Models



¹Direct use of Uemura's characterisation of GATs

It Always Works!

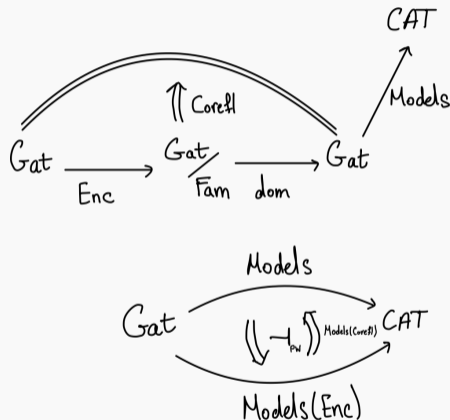
1. family encoding¹: Enc ; fully faithful
2. category of models¹: Models
3. morphisms $\text{Enc}(G) \rightarrow G$



¹Direct use of Uemura's characterisation of GATs

It Always Works!

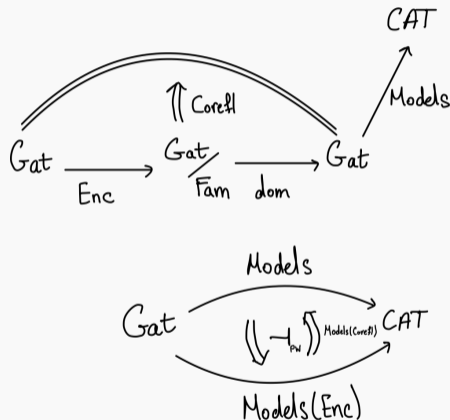
1. family encoding¹: $\text{Enc}; \text{fully faithful}$
2. category of models¹: Models
3. morphisms $\text{Enc}(G) \rightarrow G$
4. a **strict coreflection**: adjunction with $\eta = \text{Id}$



¹Direct use of Uemura's characterisation of GATs

It Always Works!

1. family encoding¹: Enc ; fully faithful
2. category of models¹: Models
3. morphisms $\text{Enc}(G) \rightarrow G$
4. a **strict coreflection**: adjunction with $\eta = \text{Id}$
 \Rightarrow the **right** adjoint applied to the initial model of $\text{Enc}(G)$ is the initial model of G



¹Direct use of Uemura's characterisation of GATs

We want

$$\begin{array}{ccc} \text{Models}(G) & \xrightarrow{\quad} & \text{Models}(\text{Enc}(G)) \\ & \perp & \\ & \xleftarrow{\quad} & \\ & \text{Models}(\text{Corefl}_G) & \end{array}$$

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

Models of the Reduced GAT

We define the family functor $F_G : \text{Models}(G) \rightarrow \text{Fam}^1$.

Theorem

Given G with family functor $F_G : \text{Models}(G) \rightarrow \text{Fam}$, we have $\text{Models}(\text{Enc}(G))$ is iso to the category whose objects consist of:

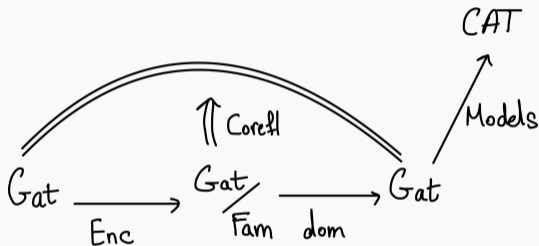
1. a model $M \in \text{Models}(G)$
2. a family (U', E')
3. a “cartesian” morphism of families of the form

$$F_G(M) \rightarrow (U', E')$$

¹Using 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 $p_G : (A : \text{Set}, a : A) \rightarrow (A : \text{Set})$, is **bi-initial** among categories with **finite limits** and a choice of an “exponentiable morphism”².

²exponentiable 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 $p_G : (A : \text{Set}, a : A) \rightarrow (A : \text{Set})$, is **bi-initial** among categories with **finite limits** and a choice of an “exponentiable morphism”².

Theorem

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

²exponentiable object in a slice

Key consequence: to define a functor out of \mathbf{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

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

mapping each theory G to its **category of models** $\text{Models}(G)$.

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

Bi-initiality then yields a functor:

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

Example (Family Encoding)

Let's define a functor

$$\text{Enc} : \text{Gat} \longrightarrow \text{Gat}/\text{Fam}$$

mapping G to $\text{Enc}(G)$, **the family encoded GAT** with sorts $(U : \text{Set}, EI : U \rightarrow \text{Set})$.





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

$$(U : \text{Set}, EI : U \rightarrow \text{Set}, A : U, a : EI A) \longrightarrow (U : \text{Set}, EI : U \rightarrow \text{Set}, A : U).$$

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

1. Adapting Uemura's universal property for FinGat to Gat
 - Our main technical tool
2. Family encoding functor $\text{Enc} : \text{Gat} \rightarrow \text{Gat}/\text{Fam}$, mapping any GAT to a family GAT with sorts $U : \text{Set}$ and $El : U \rightarrow \text{Set}$
3. Characterisation of $\text{Models}(\text{Enc}(G))$ as a kind of comma category " $F_G/cFam$ "
4. Consequence of point 3: a strict coreflection $\text{Models}(G) \begin{array}{c} \xrightarrow{\quad} \\ \perp \\ \xleftarrow{\quad} \end{array} \text{Models}(\text{Enc}(G))$
5. Proving the family encoding functor is fully faithful
 - For this we show the existence of initial models

Thank you!

-  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>.
-  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 $p : Y \rightarrow X$ (i.e. the pullback functor $p^* : \mathcal{C}/X \rightarrow \mathcal{C}/Y$ 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 $p_G : (A : \text{Set}, a : A) \rightarrow (A : \text{Set})$, is **bi-initial** in the 2-category of CartExp-categories: for any CartExp-category (C, p') there exists a CartExp-functor $\text{FinGat} \rightarrow C$ sending p_G to p' , unique up to unique isomorphism.

Models of the Reduced GAT: Key Ingredient for Coreflection

We define the family functor $F_G : \text{Models}(G) \rightarrow \text{Fam}$ using bi-initiality of Gat .

Theorem

Given G with family functor $F_G : \text{Models}(G) \rightarrow \text{Fam}$, we have $\text{Models}(\text{Enc}(G))$ is iso to the category whose objects are “cartesian” morphisms of families of the form

$$F_G(M) \rightarrow (U', El')$$

for some model $M \in \text{Models}(G)$ and family (U', El') .

A morphisms of families $f : (U_M, El_M) \rightarrow (U', El')$ is cartesian in the sense that $El'(f(u)) = El_M(u)$ for each $u \in U_M$

The Coreflection

Now that we have $\text{Models}(\text{Enc}(G)) \cong F_G /_c \text{Fam}$, we can define the coreflection.

$$\begin{array}{ccc} \text{Models}(G) & \xrightarrow{\quad} & \text{Models}(\text{Enc}(G)) \\ & \perp & \\ & \xleftarrow{\quad} & \\ & \text{Models}(\text{Corefl}_G) & \end{array}$$

- The **right adjoint** $\text{Models}(\text{Corefl}_G)$ maps a cartesian morphism $F_G(M) \rightarrow (U', E')$ to its underlying model M .
- The **left adjoint** maps a model M to the identity $F_G(M) \xrightarrow{\text{id}} F_G(M)$.
- The unit is the **identity**: *strict* coreflection.
- Round-trip starting from any model of G is the identity.

Example of the Family Encoding Functor and Coreflector

Example (Enc($A : \text{Set}$))

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

$$U : \text{Set}, \quad El : U \rightarrow \text{Set}, \quad A : U.$$

Its coreflector morphism is the following morphism:

$$\text{Corefl}_G : (U : \text{Set}, El : U \rightarrow \text{Set}, A : U) \longrightarrow (A : \text{Set})$$

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

Example: Transitive Graphs

The GAT of transitive graphs:

$V : \text{Set}$

$E : V \rightarrow V \rightarrow \text{Set}$

$T : (v1\ v2\ v3 : V) \rightarrow E\ v1\ v2 \rightarrow E\ v2\ v3 \rightarrow E\ v1\ v3$

The Family Functor: A Key Ingredient

Using bi-initiality of Gat , we get the family functor $F_G : \text{Models}(G) \rightarrow \text{Fam}$.

Example ($A : \text{Set}$)

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

Example (Transitive Graphs)

For transitive graphs, F_G 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:

$$U : \text{Set}, \quad EI : U \rightarrow \text{Set}, \quad V : U, \quad E : EI V \rightarrow EI V \rightarrow U$$

$$T : \prod_{v_1, v_2, v_3 : EI V} EI(E v_1 v_2) \rightarrow EI(E v_2 v_3) \rightarrow EI(E v_1 v_3)$$

The coreflector morphism is the GAT morphism $\text{Corefl}_G : \text{Enc}(G) \rightarrow G$ which interprets:

- the sort $V : \text{Set}$ of G as the term $EI V : \text{Set}$ in $\text{Enc}(G)$,
- the sort $E : V \rightarrow V \rightarrow \text{Set}$ of G as the term $\lambda v_1 v_2. EI(E v_1 v_2) : \text{Set}$ in $\text{Enc}(G)$,
- the operator T of G as the operator T of $\text{Enc}(G)$.

Example of the Characterisation of $\text{Models}(\text{Enc}(G))$

Example (What is $\text{Models}(\text{Enc}((A : \text{Set})))$ via $F_{A:\text{Set}}/_{c}\text{Fam}$)

For $G := (A : \text{Set})$, an object of $F_G/_{c}\text{Fam}$ consists of:

- a set X ,
- a family (U', El')
- a function $f : \{*\} \rightarrow U'$ 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 $T(A : \text{Set}) = (U : \text{Set}, El : U \rightarrow \text{Set}, A : U)$.

Also note that, the round-trip $\text{Models}(G) \rightarrow \text{Models}(\text{Enc}(G)) \rightarrow \text{Models}(G)$ starting from X gives back X .

Example of the Characterisation of Models($\text{Enc}(G)$)

Example (Models($\text{Enc}(\text{Transitive Graphs})$))

For $G := \text{Transitive Graphs}$, an object of $F_G/c\text{Fam}$ consists of:

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

Giving f is choosing:oh

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

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