

Mathematical and Computational Metatheory of Second-Order Algebraic Theories

Marcelo Fiore

University of Cambridge

HoTTtest

17. II. 2022

joint work with Dima Szamotvancer

Algebraic Type Theory

[?] What are type theories?

in logic and
computer
science

— Foundational

mathematical models

— Pragmatical

formalisation and
programming

Working hypothesis:

categorical
algebra

Type theories are algebraic systems

Example: Equational Logic of Universal Algebra

[Birkhoff]

	Equational Logic
types	unstructured sorts
terms	(first order) algebraic

Example: Equational Logic of Universal Algebra

Second-Order Equational Logic

	Equational Logic	Second-Order Equational Logic
types	unstructured sorts	(first order) algebraic
terms	(first order) algebraic	second-order

conservative extension

Second-Order Algebraic Theories

[F. & Hur]
[F. & Mahmoud]

Equational/Rewriting deduction system for
languages with

type structure: (first-order) algebraic

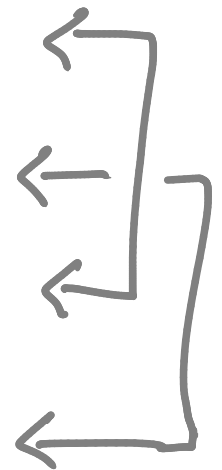
term structure: (second-order)

- variable-binding operators
- parameterised metavariables

[Frege]
[Aczel]

Plan

- Second-order algebraic theories by example.
- Mathematical foundations
vs. computer formalisation
- New mathematical modelling
& computer implementation:
 - syntax
 - binding
 - metavariables
 - substitutions
 - capture avoiding
 - meta
 - semantics



Propositional Logic

syntax PropLog

type

* : 0-ary

term

true : *

false : *

and : * * -> *

or : * * -> *

not : * -> *

theory

P : * |> and(P , P) = P

P : * |> not(not(P)) = P

P,Q : * |> not(and(P,Q)) = or(not(P) , not(Q))

First-Order Logic

syntax FOL

type

* : 0-ary

N : 0-ary

term

all : N.* \rightarrow *

ex : N.* \rightarrow *

theory

$P : N.* \mid \rightarrow \text{not}(\text{all}(x.P[x])) = \text{ex}(x. \text{not}(P[x]))$

$P, Q : N.* \mid \rightarrow \text{all}(x. \text{and}(P[x], Q[x])) = \text{and}(\text{all}(x.P[x]), \text{all}(x.Q[x]))$

$P : N.* \quad Q : * \mid \rightarrow \text{or}(\text{all}(x.P[x]), Q) = \text{all}(x. \text{or}(P[x], Q))$

$P : N.* , n : N \mid \rightarrow \text{all}(x.P[x]) = \text{and}(P[n], \text{all}(x.P[x]))$

$$F : N.* , Q : * \vdash \exists(x.F[x]) \implies Q \approx \forall(x. F[x] \implies Q)$$

$$\exists(x. F[x]) \implies Q$$

$$\equiv \neg(\exists(x.F[x])) \vee Q$$

$$\approx \forall(x. \neg F[x]) \vee Q$$

$$\approx \forall(x. \neg F[x] \vee Q) \quad (*)$$

$$\equiv \forall(x. F[x] \implies Q)$$

$$\forall(x. \neg F[x]) \vee Q \approx \forall(x. \neg F[x] \vee Q)$$

is derived from the axiom

$$P : N.* \quad Q : * \vdash \forall(x.P[x]) \vee Q \approx \forall(x. P[x] \vee Q)$$

by metasubstituting

$$\{ P := y. \neg F[y] \}$$

intuitively as follows

$$\begin{aligned} & \forall(x. P[x]) \{ P := y. \neg F[y] \} \\ & \equiv \forall(x. P[x] \{ P := y. \neg F[y] \}) \\ & \equiv \forall(x. (\neg F[y]) \{ x/y \}) \\ & \equiv \forall(x. \neg(F[y] \{ x/y \})) \\ & \equiv \forall(x. \neg F[y \{ x/y \}]) \\ & \equiv \forall(x. \neg F[x]) \end{aligned}$$

Partial Differentiation

syntax PDiff

type

R : 0-ary

term

zero	:	R			\emptyset		
add	:	R	R	->	R		\oplus
neg	:	R		->	R		\ominus
one	:	R					$\mathbb{1}$
<u>mult</u>	:	R	R	->	R		\otimes
pd	:	R.R	R	->	R		$\partial_ $

theory

$f : (R,R).R$, $g,h : R.R \mid \triangleright z : R$

$\mid - \quad \text{pd}(x.f[g[x],h[x]], z)$

$=$

$\text{add}(\text{$

$\text{mult}(\text{pd}(x.f[x,h[z]], g[z]), \text{pd}(x.g[x], z))$

$,$

$\text{mult}(\text{pd}(y.f[g[z],y], h[z]), \text{pd}(x.h[x], z))$

$\text{})$

$[R \cdot R \Vdash R] [R \Vdash R] [R \Vdash R] \triangleright [R]$

$\vdash \quad \partial_0 \alpha \langle b \langle x_0 \rangle \blacktriangleleft c \langle x_0 \rangle \rangle$

\approx_a

$(\partial \alpha \langle x_0 \blacktriangleleft c \langle x_1 \rangle \rangle \mid b \langle x_0 \rangle) \otimes (\partial_0 b \langle x_0 \rangle)$

\oplus

$(\partial \alpha \langle b \langle x_1 \rangle \blacktriangleleft x_0 \rangle \mid c \langle x_0 \rangle) \otimes (\partial_0 c \langle x_0 \rangle)$

-- Unary chain rule

$$\partial_\theta a(b(x_\theta))$$

$$\approx \langle \text{ax } \partial \text{Ch}_2 \text{ with } \langle \langle a(x_\theta) \triangleleft b(x_\theta) \triangleleft \mathbb{0} \rangle \rangle \rangle$$

$$\begin{aligned} & (\partial a(x_\theta) \mid b(x_\theta)) \otimes (\partial_\theta b(x_\theta)) \\ \oplus & (\partial a(b(x_1)) \mid \mathbb{0}) \otimes (\partial_\theta \mathbb{0}) \end{aligned}$$

$$\approx \langle \text{cong}[\text{thm } \partial \mathbb{0}] \text{ inside } (\partial a(x_\theta) \mid b(x_\theta)) \otimes (\partial_\theta b(x_\theta)) \oplus ((\partial a(b(x_1)) \mid \mathbb{0}) \otimes \circ^c) \rangle$$

$$\begin{aligned} & (\partial a(x_\theta) \mid b(x_\theta)) \otimes (\partial_\theta b(x_\theta)) \\ \oplus & (\partial a(b(x_1)) \mid \mathbb{0}) \otimes \mathbb{0} \end{aligned}$$

$$\approx \langle \text{cong}[\text{thm } \mathbb{0}X^{\otimes R}] \text{ with } \langle (\partial a(b(x_1)) \mid \mathbb{0}) \rangle \rangle \text{ inside } (\partial a(x_\theta) \mid b(x_\theta)) \otimes (\partial_\theta b(x_\theta)) \oplus \circ^c \rangle$$

$$\begin{aligned} & (\partial a(x_\theta) \mid b(x_\theta)) \otimes (\partial_\theta b(x_\theta)) \\ \oplus & \mathbb{0} \end{aligned}$$

$$\approx \langle \text{thm } \mathbb{0}U^{\otimes R} \text{ with } \langle (\partial a(x_\theta) \mid b(x_\theta)) \otimes (\partial_\theta b(x_\theta)) \rangle \rangle$$

$$(\partial a(x_\theta) \mid b(x_\theta)) \otimes \partial_\theta b(x_\theta)$$

Formal Metatheory of Second-Order Abstract Syntax

Marcelo Fiore and Dmitrij Szamozvancev. 2022. Formal Metatheory of Second-Order Abstract Syntax. *Proc. ACM Program. Lang.* 6, POPL, Article 53 (January 2022), 29 pages. <https://doi.org/10.1145/3498715>

We present a mathematically-inspired language-formalisation framework implemented in Agda. The system translates the description of a syntax signature with variable-binding operators into an intrinsically-encoded, inductive data type equipped with syntactic operations such as weakening and substitution, along with their correctness properties. The generated metatheory further incorporates metavariables and their associated operation of metasubstitution, which enables second-order equational/rewriting reasoning. The underlying mathematical foundation of the framework – initial algebra semantics – derives compositional interpretations of languages into their models satisfying the semantic substitution lemma by construction.

Applications:

- Formalisation and reasoning
 - Logic
 - Programming | calculi
- Symbolic computation
 - Deduction systems | rapid prototyping
 - Programming calculi
 - Abstract machines | formal translations

Mathematical Theory [F. & Plotkin & Turi]

Presheaf Model of Abstract Syntax with Variable Binding

- Types : T
- Category of contexts and renamings : $\mathbb{F}[T]$
- Universe of discourse : $\tau \in (\underline{\text{Set}}^{\mathbb{F}[T]})^T$

$\tau_\alpha(\Gamma) \sim$ terms of type α
in context Γ

Example: Presheaf of variables

$$V_\alpha(\Gamma) = \mathbb{F}[T](\langle \alpha \rangle, \Gamma)$$

- Combinatorial constructions

sums

$$\sum_i \tau_i$$

products

$$\prod_i \tau_i$$

powers to representables

$$\tau_\beta^{\vee_\alpha}$$

Terms of type β in a context extended by α

$$\tau_\beta^{\vee_\alpha}(\Gamma) = \tau_\beta(\alpha.\Gamma)$$

- Combinatorial constructions

sums

$$\bigsqcup_i \tau_i$$

products

$$\prod_i \tau_i$$

powers to representables

$$\tau_\beta^{\vee_\alpha}$$

Terms of type β in a context extended by α

$$\tau_\beta^{\vee_\alpha}(\Gamma) = \tau_\beta(\alpha.\Gamma)$$

Example:

$$t ::= x \mid t_1 @ t_2 \mid \lambda x. t$$

$$\tau \cong V + \tau \times \tau + \tau^\vee$$

Initial-Algebra Semantics

[ADJ]

Syntax signatures \rightsquigarrow Polynomial functors

Abstract syntax \rightsquigarrow Initial algebras

- universal representation
- compositional semantics
- induction principles

Initial-Algebra Semantics

[ADJ]

Syntax signatures \rightsquigarrow Polynomial functors

Abstract syntax \rightsquigarrow Initial algebras

- universal representation
- compositional semantics
- induction principles

Thesis

- * Type-theoretic rules are syntactic descriptions of polynomial diagrams.
- * The abstract syntax is the initial algebra for the associated polynomial functor.

Framework

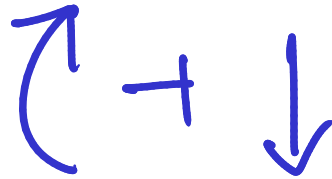
syntax

second-order
signature

$$\Sigma \rightsquigarrow P_{\Sigma} \hookrightarrow (\underline{\text{Set}}^{\mathbb{F}[\tau]})^T$$

$P_{\Sigma} - \underline{\text{Alg}}$

semantics



$$\hookrightarrow (\underline{\text{Set}}^{\mathbb{F}[\tau]})^T$$

T_{Σ}

term model.

Computer Implementation Approaches

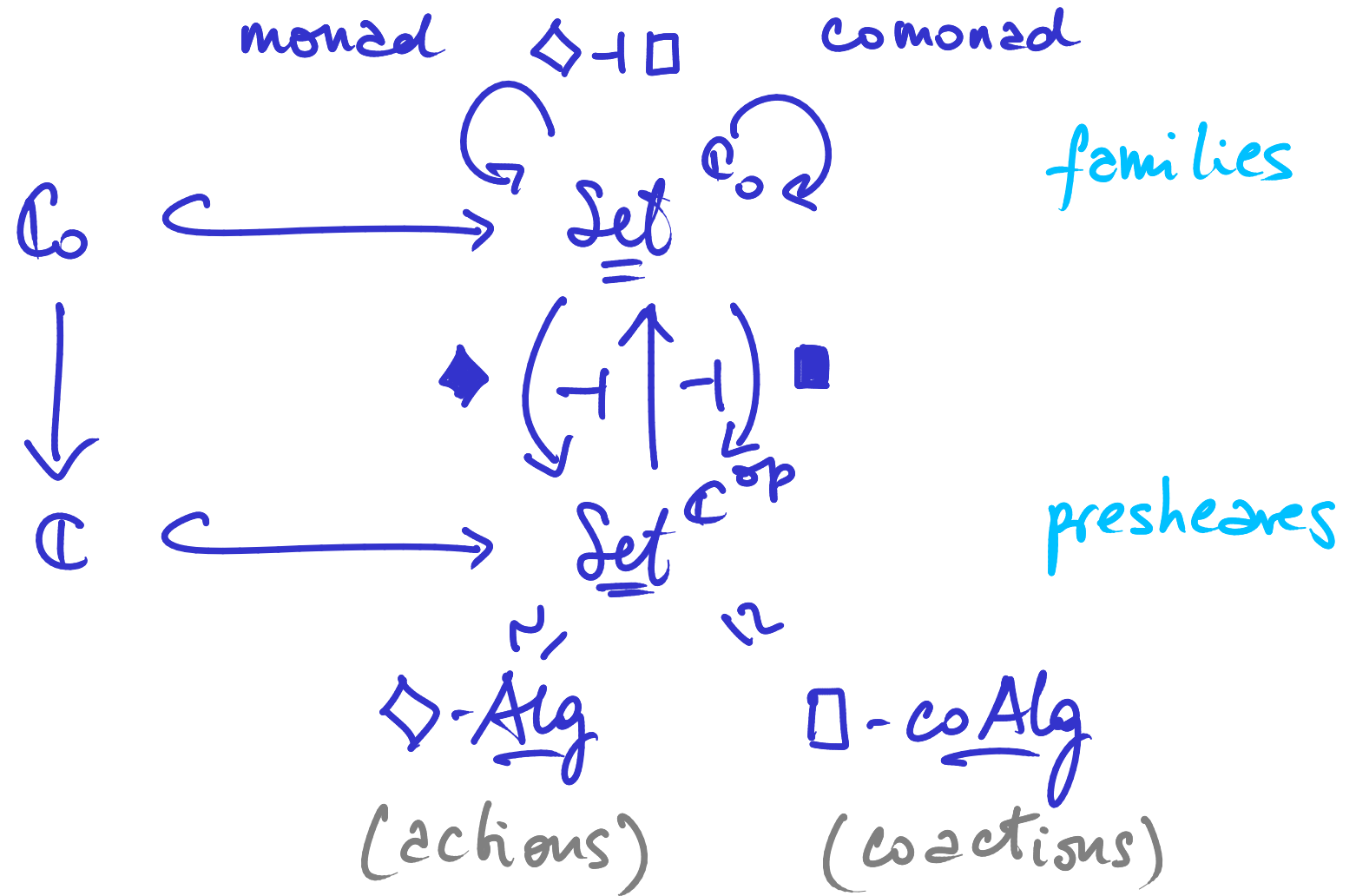
- In programming languages
 - Ad hoc
 - No correctness guarantees.
- In proof assistants
 - Strong invariants for correctness
 - computation

Mathematics vs. Proof Assistant

- formalise the mathematical model in the proof assistant
 - computationally problematic
- adapt the mathematics to the proof assistant

$$\begin{array}{ccc} & \text{TENSION} & \\ \text{presheaves} & \longleftrightarrow & \text{families} \\ \left(\underline{\text{Set}}^{\#[\tau]} \right)^T & & \left(\underline{\text{Set}}^{T*} \right)^T \end{array}$$

Adjoint Modalities



$$\blacklozenge F(\Gamma) = \sum_{\Delta} F(\Delta) * \mathbb{C}(\Gamma, \Delta)$$

$$\blacksquare F(\Gamma) = \prod_{\Delta} \mathbb{C}(\Delta, \Gamma) \Rightarrow \bar{F}\Delta$$

Presheaves as Indexed Families with Algebraic Model Structure in Proof Assistants?

Internal languages for presheaves
compiled to families via adjoint
modalities?

Families $(\underline{\text{Set}}^{T^*})^T$ as universe of discourse

- Names: $N_\alpha(\Gamma) = (\langle \alpha \rangle \equiv \Gamma)$
- Combinatorial constructions

$$\coprod_i F_i, \quad \prod_i F_i$$

Families $(\underline{\text{Set}}^{T*})^T$ as universe of discourse

- Names: $N_\alpha(\Gamma) = (\langle \alpha \rangle \equiv \Gamma)$
- Combinatorial constructions

$$\coprod_i F_i, \quad \prod_i F_i$$

Day
convolution

$$(F \otimes g)(\Gamma) = \sum_{\Gamma = \Gamma_1 + \Gamma_2} F(\Gamma_1) \times g(\Gamma_2)$$

$$(F \multimap g)(\Gamma) = \prod_{\Delta} F(\Delta) \Rightarrow g(\Gamma + \Delta)$$

Calculus

- $V = \blacklozenge N$ variables are freely generated from names

$$I = |V| \quad \text{indices}$$

- $N_\alpha \multimap F_\beta = F_\beta(\alpha \cdot -)$ context extension

$$\bullet \quad |P^{\blacklozenge X}| = X \multimap |P|$$

$$\bullet \quad \blacklozenge(F) \times \blacklozenge(G) = \blacklozenge(F \otimes G)$$

$$\bullet \quad (\blacksquare F)^{\mathbb{Z}} = \blacksquare(F^{|Z|})$$

Initial Algebra Semantics

Example: λ -calculus $t ::= x \mid \lambda x.t_2 \mid \lambda x.t$

$$\mathcal{T} \cong V + \mathcal{T} \times \mathcal{T} + \mathcal{T}^V$$

in presheaves



$$|\mathcal{T}| \cong |V| + |\mathcal{T}| \times |\mathcal{T}| + |\mathcal{T}|^{\blacklozenge N}$$

$$\cong I + |\mathcal{T}| \times |\mathcal{T}| + N \multimap |\mathcal{T}|$$



$$\mathcal{F} \cong I + \mathcal{F} \times \mathcal{F} + N \multimap \mathcal{F}$$

in families

$$\begin{array}{ccc}
 (\underline{\text{Set}}^{\mathbb{F}[\tau]})^T & \xrightarrow{P_\Sigma} & (\underline{\text{Set}}^{\mathbb{F}[\tau]})^T \\
 \downarrow & & \downarrow \\
 (\underline{\text{Set}}^{\tau*})^T & \xrightarrow{P_\Sigma^\downarrow} & (\underline{\text{Set}}^{\tau*})^T
 \end{array}$$

Initial P_Σ^\downarrow -algebras lift to initial P_Σ -algebras

Initial-Algebras Lifting Theorem

$$\begin{array}{ccc}
 \mathcal{S}\text{-coAlg} & \xrightarrow{F_\lambda} & \mathcal{S}\text{-coAlg} \\
 \downarrow & & \downarrow \\
 \mathcal{E} & \xrightarrow{F} & \mathcal{E}
 \end{array}$$

Initial F -algebras lift to initial F_λ -algebras

$$\begin{array}{ccccc}
 FA & \xrightarrow{\alpha \text{ initial}} & A & & \\
 \downarrow & & \downarrow \exists! & \text{coalgebra} & \\
 FSA & \xrightarrow{\lambda} SFA & \xrightarrow{S\alpha} & SA & \text{structure}
 \end{array}$$

syntax STLC | Λ

type

N : 0-ary

\rightarrow : 2-ary

term

app : $\alpha \rightarrow \beta$ α \rightarrow β | $_ \$ _$

lam : $\alpha.\beta$ \rightarrow $\alpha \rightarrow \beta$ | $\lambda _$

data Λ : Family_s where

var : $\mathcal{J} \rightarrow \Lambda$

$_ \$ _$: Λ ($\alpha \rightarrow \beta$) Γ \rightarrow Λ α Γ \rightarrow Λ β Γ

$\lambda _$: Λ β ($\alpha \bullet \Gamma$) \rightarrow Λ ($\alpha \rightarrow \beta$) Γ

automatically
generated
intrinsically-typed
encoding

[Altenkirch & Reus,
Benton et al,
Allass et al]

Substitution

Idea: metoperation

$$\mathcal{E}_\alpha(\Gamma) \times \prod_{\gamma \in \Gamma} \mathcal{E}_\gamma(\Delta) \longrightarrow \mathcal{E}_\alpha(\Delta)$$

subject to axioms.

Substitution

Idea: meta operation

$$\mathcal{E}_\alpha(\Gamma) \times \prod_{\gamma \in \Gamma} \mathcal{E}_\gamma(\Delta) \longrightarrow \mathcal{E}_\alpha(\Delta)$$

subject to axioms.

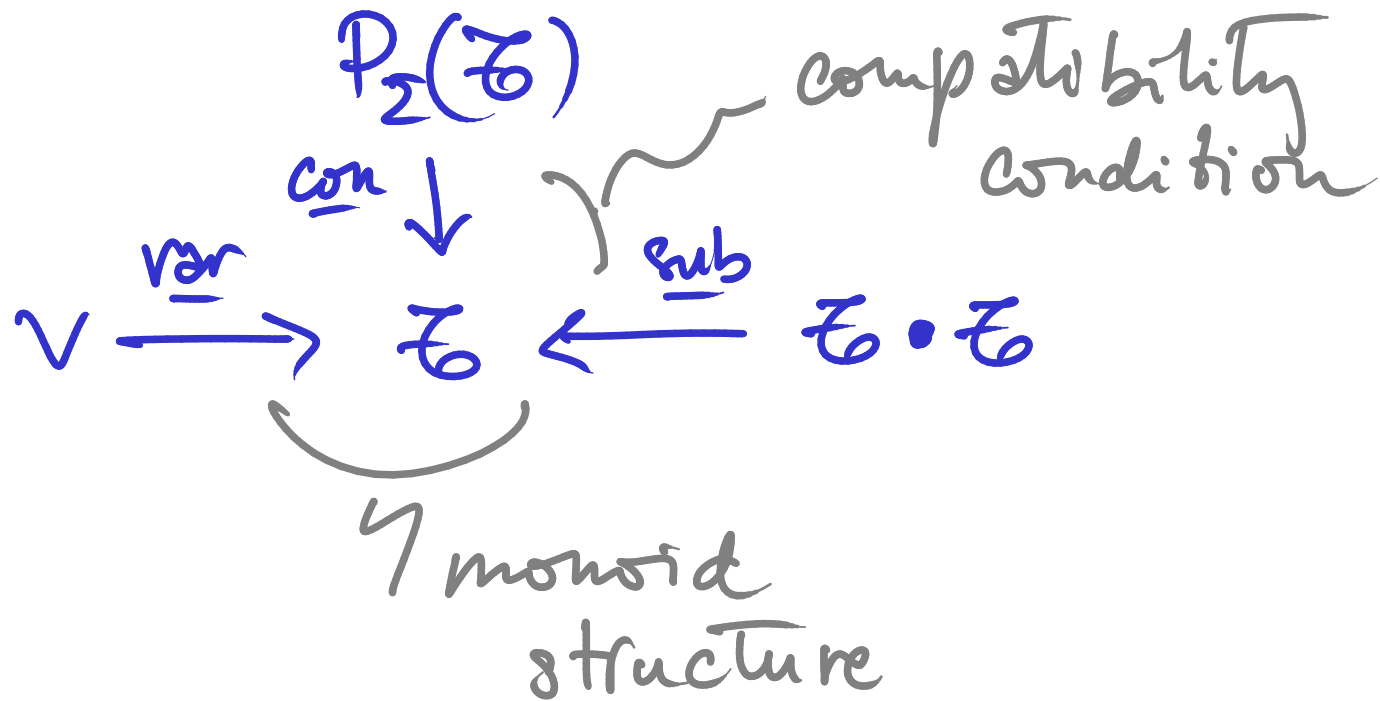
Presheaf model: monoid structure

$$\mathcal{E} \bullet \mathcal{E} \longrightarrow \mathcal{E}$$

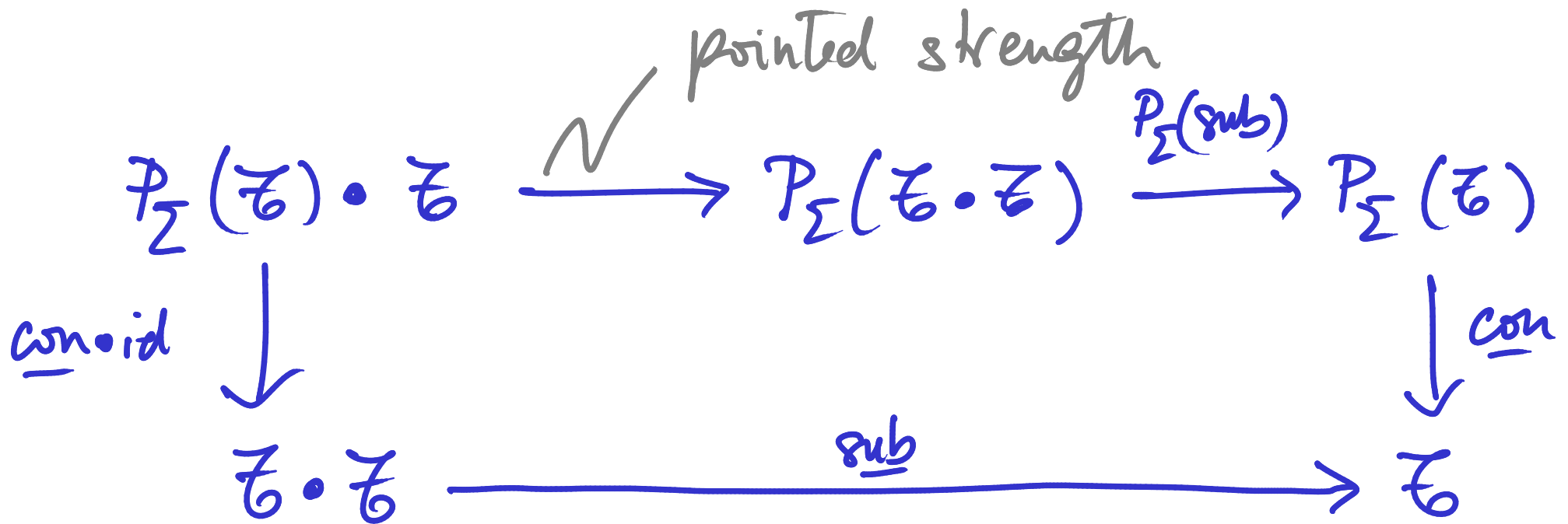
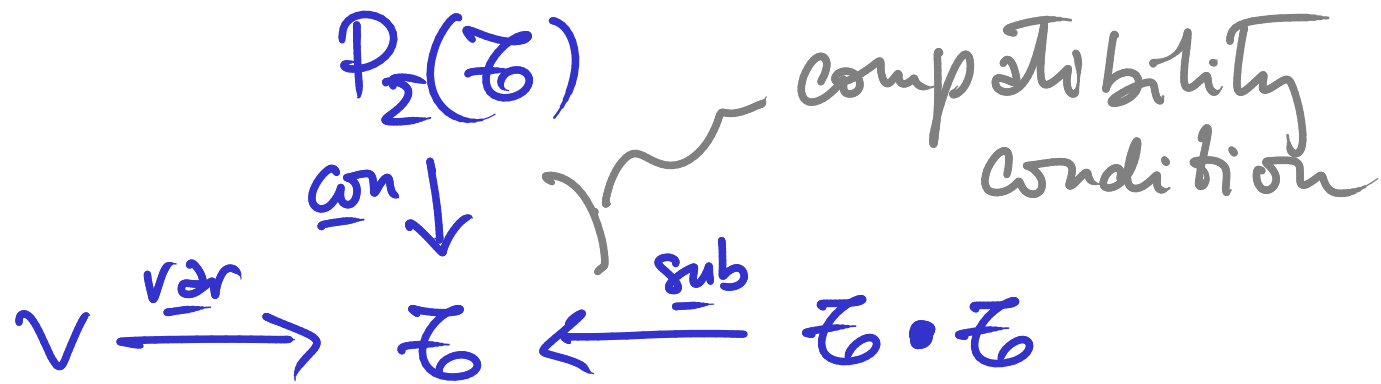
for a substitution tensor product

$$(\mathcal{E} \bullet \mathcal{S})_\alpha(\Delta) = \int^\Gamma \mathcal{E}_\alpha(\Gamma) \times \prod_{\gamma \in \Gamma} \mathcal{S}_\gamma(\Delta)$$

Syntax with Substitution



Syntax with Substitution



Thm [H2man2, F.]

$$\begin{array}{ccc}
 & \Sigma\text{-Mon} & \\
 & \uparrow \downarrow & \\
 \mathcal{M} & \hookrightarrow & (\text{Set}^{\mathbb{F}[\tau]})^{\tau}
 \end{array}$$

where the free Σ -monoid on \mathcal{X} is an initial:

$$\mathcal{M}(\mathcal{X}) \cong V + \mathcal{X} \bullet \mathcal{M}(\mathcal{X}) + P_{\Sigma}(\mathcal{M}\mathcal{X})$$

- Provides inductively defined abstract syntax with variable binding operators and parameterised metavariables.

$$\mathcal{M}(\mathcal{X}) \cong V + \mathcal{X} \bullet \mathcal{M}(\mathcal{X}) + P_{\Sigma}(\mathcal{M}\mathcal{X})$$

$$t ::= x \mid M[t_1, \dots, t_n] \mid f(\dots, \vec{x}.t', \dots)$$

- Provides a provably-correct inductively-defined substitution operation, automatically preserved by semantic interpretations

Generalises type-theoretic/semantic practices

Free Algebras with Substitution in Families

- Skew substitution tensor product [Borthelle
& Hirschowitz
& Lefont]

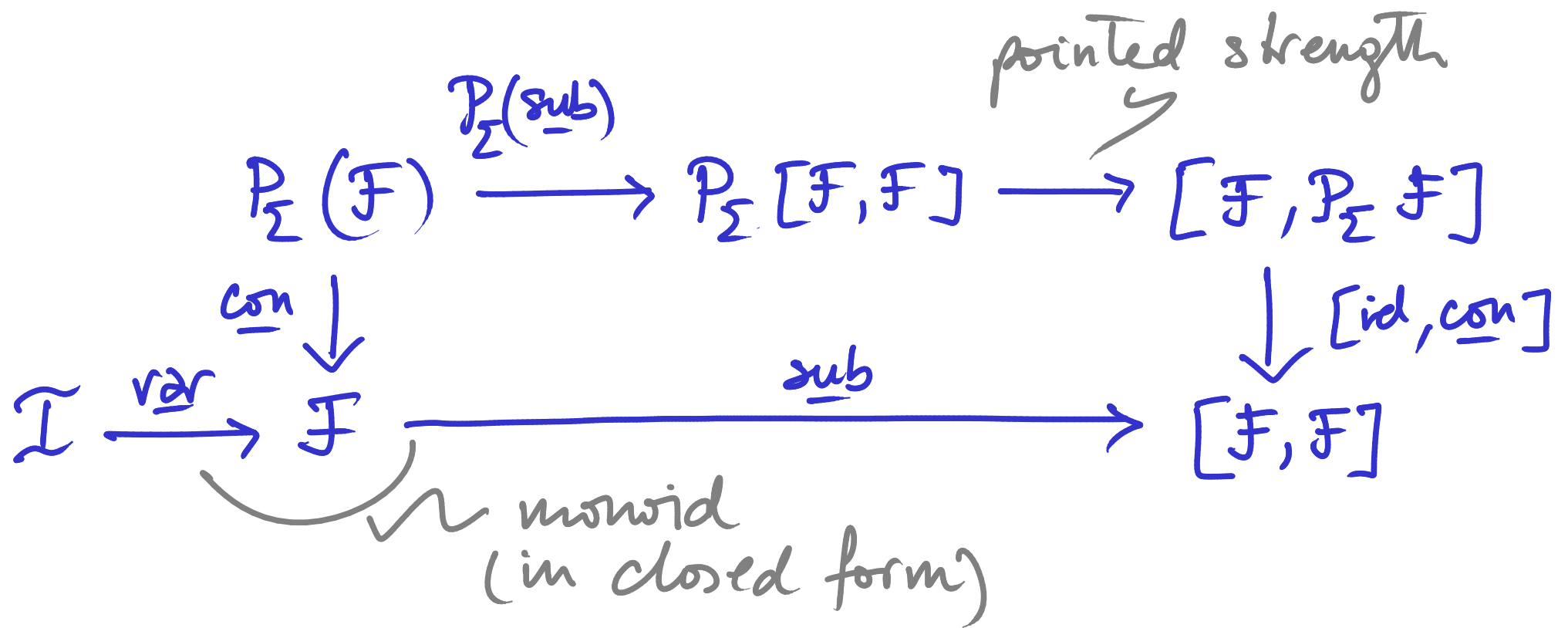
$$(g \cdot F)_\alpha(\Delta) = \sum_{\Gamma} g_\alpha(\Gamma) \times \prod_{\gamma \in \Gamma} F_\gamma(\Delta)$$

(with unit the family of indices)

NB: Monoids are, equivalently, abstract clones

Syntax with Substitution in Families

$$[\mathbb{F}, g]_{\alpha}(\Gamma) = \prod_{\Delta} \left(\prod_{\delta \in \Gamma} \mathbb{F}_{\delta}(\Delta) \right) \Rightarrow g_{\alpha}(\Delta)$$



Thm: For every family $\mathcal{X} \in (\underline{\text{Set}}^{\tau*})^{\tau}$, the initial algebra $\mathcal{M}_{\Sigma}(\mathcal{X})$ with structure

$$\begin{cases} \underline{\text{var}}: I \rightarrow \mathcal{M}_{\Sigma}(\mathcal{X}) \\ \underline{\text{mvar}}: \mathcal{X} \rightarrow [\mathcal{M}_{\Sigma}(\mathcal{X}), \mathcal{M}_{\Sigma}(\mathcal{X})] \\ \underline{\text{con}}: P_{\Sigma}(\mathcal{X}) \rightarrow \mathcal{X} \end{cases}$$

is the free Σ -monoid on \mathcal{X} .

syntax STLC | Λ

type

N : 0-ary

\rightarrow : 2-ary

term

app : $\alpha \rightarrow \beta$ α \rightarrow β | $_ \$ _$

lam : $\alpha . \beta$ \rightarrow $\alpha \rightarrow \beta$ | $\lambda _$

module Λ :Terms (\mathfrak{x} : Family_s) where

data Λ : Family_s where

var : \mathbb{J} \rightarrow Λ

mvar : \mathfrak{x} α Π \rightarrow Sub Λ Π Γ \rightarrow Λ α Γ

$_ \$ _$: Λ ($\alpha \rightarrow \beta$) Γ \rightarrow Λ α Γ \rightarrow Λ β Γ

$\lambda _$: Λ β ($\alpha \bullet \Gamma$) \rightarrow Λ ($\alpha \rightarrow \beta$) Γ

automatically
generated
intrinsically-typed
encoding with
metavariables

proof ingredients:

- substitution operation

$$\underline{\text{sub}}: \mathcal{M}_Z(X) \xrightarrow{\quad} [\mathcal{M}_Z(X), \mathcal{M}_Z(X)]$$

induced by initiality

requires \square -coalgebra structure on $\mathcal{M}_Z(X)$
[recall the initial-algebra lifting thm]

proof ingredients:

- substitution operation

$$m_2(x) \xrightarrow{\quad} [m_2(x), m_2(x)]$$

induced by initiality

requires \square -coalgebra structure on $m_2(x)$
[recall the initial-algebra lifting thm]

$$\begin{array}{ccc} I \cdot m(x) & & \\ \downarrow & \searrow & \\ m(x) \cdot m(x) & \xrightarrow{\text{sub}} & m(x) \end{array}$$

$$\begin{array}{ccccc} P(m(x) \cdot m(x)) & \rightarrow & P(m(x) \cdot m(x)) & \xrightarrow{P_{\text{sub}}} & P(m(x)) \\ \downarrow & & & & \downarrow \\ m(x) \cdot m(x) & \xrightarrow{\text{sub}} & & & m(x) \end{array}$$

$$\begin{array}{ccccc} (x \cdot m(x)) \cdot m(x) & \rightarrow & x \cdot (m(x) \cdot m(x)) & \xrightarrow{x \cdot \text{sub}} & x \cdot m(x) \\ \downarrow & & & & \downarrow \\ m(x) \cdot m(x) & \xrightarrow{\text{sub}} & & & m(x) \end{array}$$

proof ingredients:

- substitution operation

$$M_Z(X) \rightarrow [M_Z(X), M_Z(X)]$$

induced by initiality

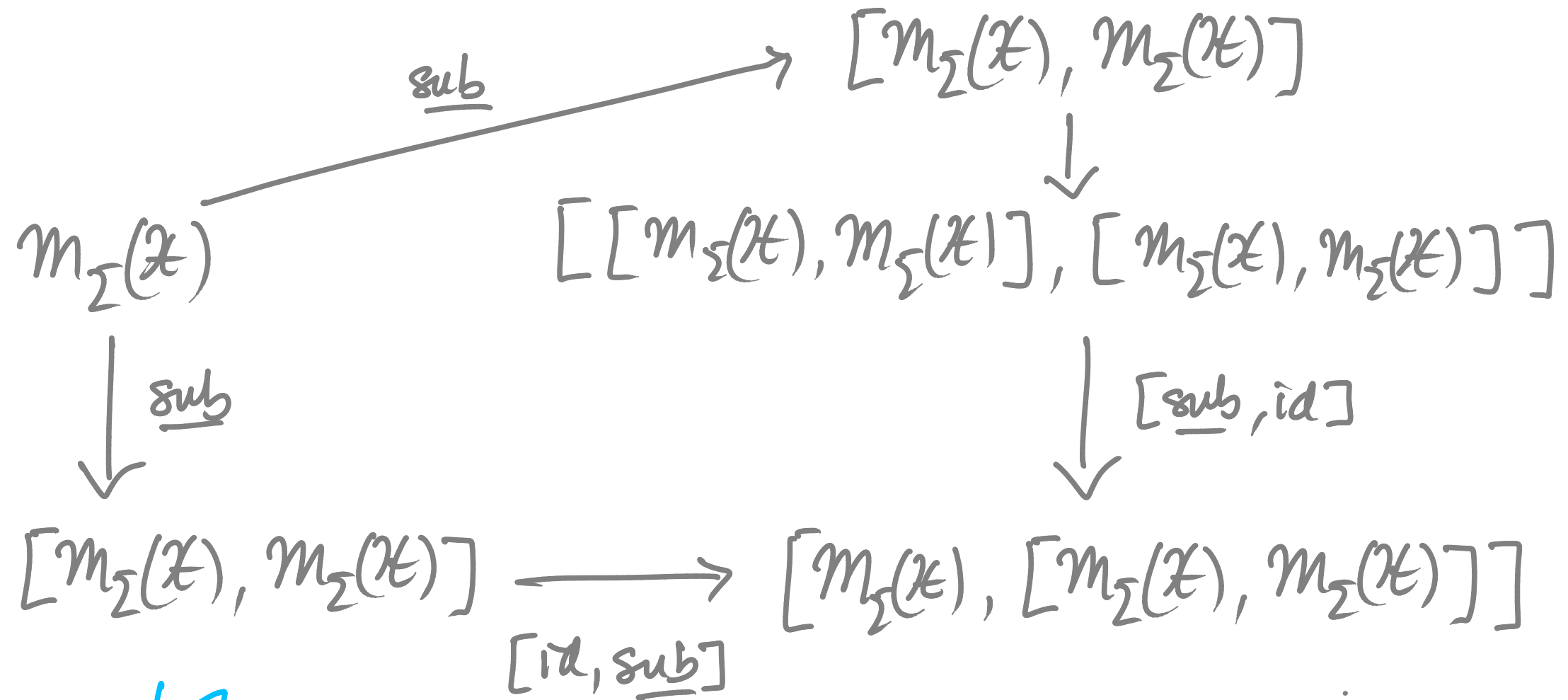
requires \square -coalgebra structure on $M_Z(X)$
[recall the initial-algebra lifting thm]

derived from general traversals [McBride et al]

$$M_Z(X) \rightarrow [P, \alpha]$$

parameterised
semantic
models

- substitution laws by initiality



crucially using The Theory of traversals

Metasubstitution in Presheaves

Thm [F.]: In the presheaf model,

\mathcal{M}_Σ is an enriched monad

\leadsto induces an (inductively-defined
provably correct internal)
metasubstitution operation

$$\underline{m}_{\text{sub}}: \mathcal{M}_\Sigma(x) \times (\mathcal{M}_\Sigma(y))^x \longrightarrow \mathcal{M}_\Sigma(y)$$

Metasubstitution in Families

Linear internalisation

$$\underline{\text{msub}}: \mathcal{M}_Z(x) \longrightarrow (x \multimap \mathcal{M}_Z(y)) \multimap \mathcal{M}_Z(y)$$

↙
may be induced by inibridality

In elementary terms:

$$\mathcal{M}_Z(x)(\Gamma_1) \longrightarrow \left(\prod_{\Delta} x(\Delta) \Rightarrow \mathcal{M}_Z(y)(\Delta + \Gamma_2) \right) \Rightarrow \mathcal{M}_Z(y)(\Gamma_1 + \Gamma_2)$$

Meta substitution recursion

- $I \rightarrow (\lambda \rightarrow m_\Sigma y) \rightarrow m_\Sigma(y)$

$$i \mapsto \sigma \mapsto \underline{\text{var}}(i)$$

- $P_\Sigma((\lambda \rightarrow m_\Sigma y) \rightarrow m_\Sigma y) \rightarrow (\lambda \rightarrow m_\Sigma y) \rightarrow m_\Sigma y$

linear strength \searrow

$$(\lambda \rightarrow m_\Sigma y) \rightarrow P_\Sigma(m_\Sigma y)$$

- $\lambda \xrightarrow{\text{rec}} [(\lambda \rightarrow m_\Sigma y) \rightarrow m_\Sigma y, (\lambda \rightarrow m_\Sigma y) \rightarrow m_\Sigma y]$

$$m \mapsto \varepsilon \mapsto \sigma \mapsto \underline{\text{sub}}(\sigma m, [\varepsilon \sigma, \underline{wk}])$$

$$\underline{msub} (\underline{mvar}(m, t)) \sigma$$

$$= \underline{sub} (\sigma(m), [\underline{msub} \ t \ \sigma, \underline{wk}])$$

$$\begin{array}{ccc}
 x \cdot m(x) & \xrightarrow{\underline{rec} \cdot \underline{msub}} & [(x \multimap my) \multimap my, (x \multimap my) \multimap my] \\
 \downarrow \underline{mvar} & & \bullet (x \multimap my) \multimap my \\
 & & \downarrow \underline{eval} \\
 m(x) & \xrightarrow{\underline{msub}} & my \multimap (x \multimap my)
 \end{array}$$

Second-order Equational Logic

variable

$\alpha \ \beta \ \gamma : T$
 $\Gamma \ \Delta \ \Pi : \text{Ctx}$
 $\mathfrak{M} \ \mathfrak{N} : \text{MCtx}$

-- Second-order equational logic

module EqLogic (_▷_ _⊢_ ≈_a_ : $\forall \mathfrak{M} \underline{\Gamma} \{\alpha\} \rightarrow (\mathfrak{M} \triangleright T) \ \alpha \ \underline{\Gamma} \rightarrow (\mathfrak{M} \triangleright T) \ \alpha \ \underline{\Gamma} \rightarrow \text{Set}$) where

data _▷_ _⊢_ ≈ : $(\mathfrak{M} : \text{MCtx})\{\alpha : T\}(\underline{\Gamma} : \text{Ctx}) \rightarrow (\mathfrak{M} \triangleright T) \ \alpha \ \underline{\Gamma} \rightarrow (\mathfrak{M} \triangleright T) \ \alpha \ \underline{\Gamma} \rightarrow \text{Set}_1$ where

ax : $\{t \ s : (\mathfrak{M} \triangleright T) \ \alpha \ \underline{\Gamma}\}$
 $\rightarrow \mathfrak{M} \triangleright \underline{\Gamma} \vdash t \approx_a s$

 $\rightarrow \mathfrak{M} \triangleright \underline{\Gamma} \vdash t \approx s$

eq : $\{t \ s : (\mathfrak{M} \triangleright T) \ \alpha \ \underline{\Gamma}\}$
 $\rightarrow t \equiv s$

 $\rightarrow \mathfrak{M} \triangleright \underline{\Gamma} \vdash t \approx s$

sy : $\{t \ s : (\mathfrak{M} \triangleright T) \ \alpha \ \underline{\Gamma}\}$
 $\rightarrow \mathfrak{M} \triangleright \underline{\Gamma} \vdash t \approx s$

 $\rightarrow \mathfrak{M} \triangleright \underline{\Gamma} \vdash s \approx t$

tr : $\{t \ s \ u : (\mathfrak{M} \triangleright T) \ \alpha \ \underline{\Gamma}\}$
 $\rightarrow \mathfrak{M} \triangleright \underline{\Gamma} \vdash t \approx s$
 $\rightarrow \mathfrak{M} \triangleright \underline{\Gamma} \vdash s \approx u$

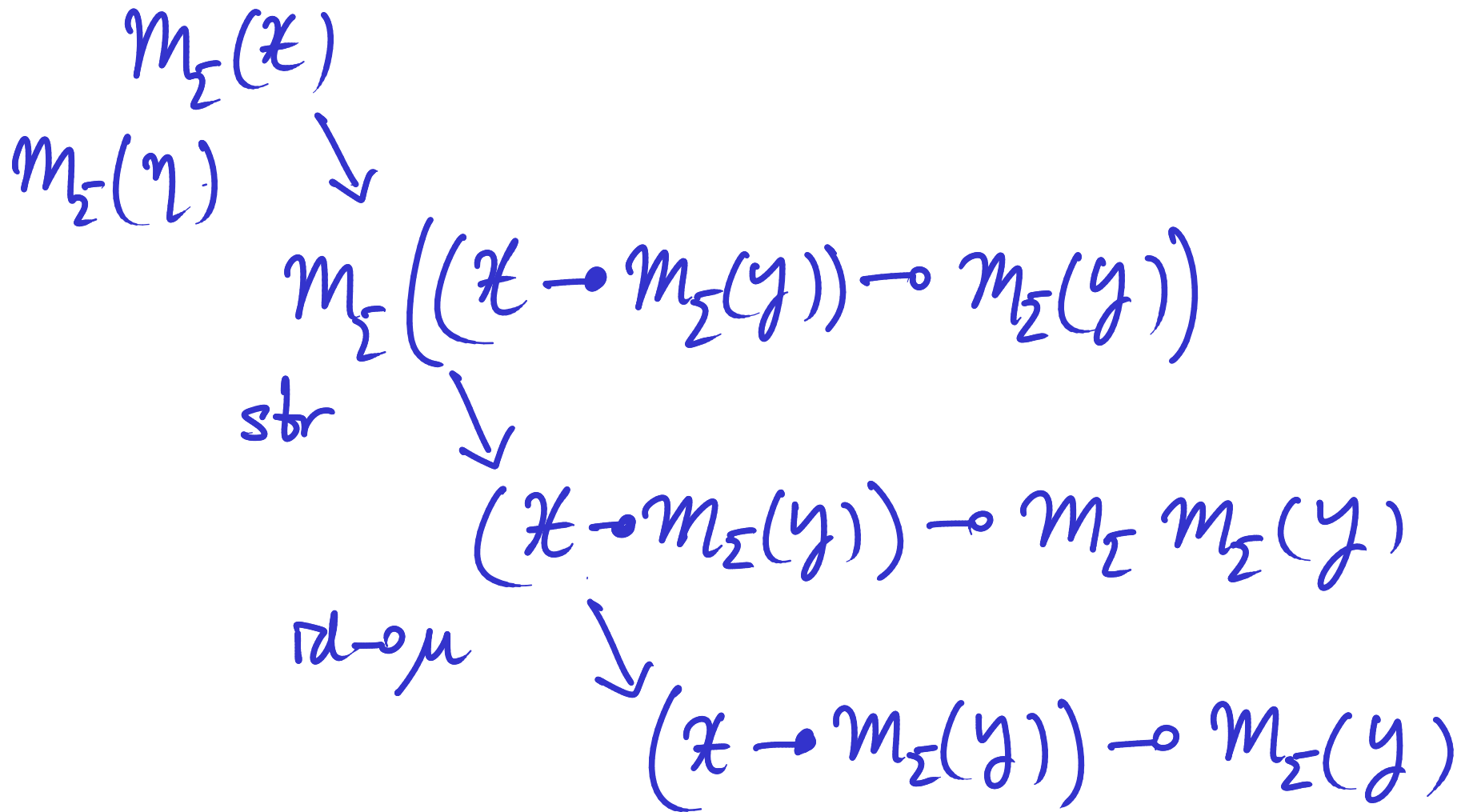
 $\rightarrow \mathfrak{M} \triangleright \underline{\Gamma} \vdash t \approx u$

□ms : $\{t \ s : (\mathfrak{M} \triangleright T) \ \alpha \ \underline{\Gamma}\}$

$\rightarrow \mathfrak{M} \triangleright \underline{\Gamma} \vdash t \approx s$
 $\rightarrow (\rho : \underline{\Gamma} \rightsquigarrow \underline{\Delta})$
 $\rightarrow (\zeta \ \xi : \text{MSub} \ \underline{\Delta} \ \mathfrak{M} \ \mathfrak{N})$
 $\rightarrow (\forall \{\tau \ \Pi\} (m : \Pi \Vdash \tau \in \mathfrak{M}) \rightarrow \mathfrak{N} \triangleright (\Pi + \underline{\Delta}) \vdash (\text{ixl} \ \zeta \ m) \approx (\text{ixl} \ \xi \ m))$

 $\rightarrow \mathfrak{N} \triangleright \underline{\Delta} \vdash (\text{□msub} \ t \ \rho \ \zeta) \approx (\text{□msub} \ s \ \rho \ \xi)$

► The laws of metasubstitution are approached by a decomposition:

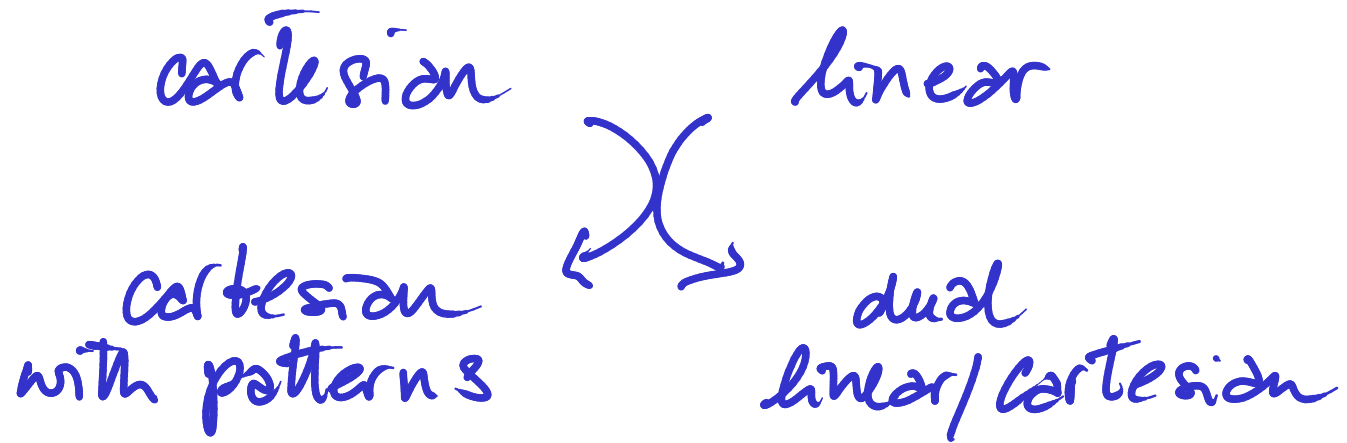


Conclusion

- Automatic generation of generic:
 - second-order abstract syntax
 - provably-correct substitution and meta substitution operations
 - algebraic models with compositional interpretations
- Agda implementation
 - for deduction and computation
 - mathematically inspired

Directions

- Application case studies
 - Simply-Typed Contextual Model Type Theory
[Wanerski & Pfenning & Pientke]
- Linearity



- Reflection
- Polymorphism
- Type dependency