Contextual Algebraic Theories:
Generic Boilerplate Beyond Abstraction
(Extended Abstract / Work in Progress)

Andreas Nuyts

KU Leuven, Belgium

TyDe '22
Ljubljana, Slovenia
Sep 11, 2022

Boilerplate

For YOUR language, you may want:

- Admissibility of renaming & substitution,
- Soundness,
- Canonicity,
- Normalization,
- Decidability of equality,
- An interpreter,
- Compilers,
  - Desugarings,
- A pretty-printer.

Every time, you will face **BOILERPLATE**!

Boilerplate

For YOUR language, you may want:

- Admissibility of renaming & substitution,
- Soundness,
- Canonicity,
- Normalization,
- Decidability of equality,
- An interpreter,
- Compilers,
  - Desugarings,
- A pretty-printer.

Every time, you will face **BOILERPLATE**!

Boilerplate

For YOUR language, you may want:

- Admissibility of renaming & substitution,
- Soundness,
- Canonicity,
- Normalization,
- Decidability of equality,
- An interpreter,
- Compilers,
  - Desugarings,
- A pretty-printer.

Every time, you will face **BOILERPLATE**!

Boilerplate

For YOUR language, you may want:

- Admissibility of renaming & substitution,
- Soundness,
- Canonicity,
- Normalization,
- Decidability of equality,
- An interpreter,
- Compilers,
  - Desugarings,
- A pretty-printer.

Every time, you will face **BOILERPLATE**!

Boilerplate

For YOUR language, you may want:

- Admissibility of renaming & substitution,
- Soundness,
- Canonicity,
- Normalization,
- Decidability of equality,
- An interpreter,
- Compilers,
    - Desugarings,
- A pretty-printer.

Every time, you will face **BOILERPLATE**!

Boilerplate

For YOUR language, you may want:

- Admissibility of renaming & substitution,
- Soundness,
- Canonicity,
- Normalization,
- Decidability of equality,
- An interpreter,
- Compilers,
    - Desugarings,
- A pretty-printer.

Every time, you will face **BOILERPLATE**!

Boilerplate

For YOUR language, you may want:

- Admissibility of renaming & substitution,
- Soundness,
- Canonicity,
- Normalization,
- Decidability of equality,
- An interpreter,
- Compilers,
    - Desugarings,
- A pretty-printer.

Every time, you will face **BOILERPLATE**!

Boilerplate in Menkar

	Π	λ	app	Σ	pair	fst	snd	...
Renaming								
Substitution								
Scope-checker								
Pretty-printer								
Type-checker								
Def. Equality-checker								
Synt. Equality-checker								
Metavariable-solver								
WHNormalizer								

Still not feasible to maintain / develop further.

Boilerplate in Menkar

	Π	λ	app	Σ	pair	fst	snd	...
Renaming	deriving Functor **(BAD PERFORMANCE)**							
Substitution	deriving Monad generically **(BAD PERFORMANCE)**							
Scope-checker								
Pretty-printer								
Type-checker								
Def. Equality-checker								
Synt. Equality-checker								
Metavariable-solver								
WHNormalizer								

Still not feasible to maintain / develop further.

Boilerplate in Menkar

		Π	λ	app	Σ	pair	fst	snd	...
Renaming		deriving Functor **(BAD PERFORMANCE)**							
Substitution		deriving Monad generically **(BAD PERFORMANCE)**							
Scope-checker									
Pretty-printer									
Type-checker	Analyzer								
Def. Equality-checker									
Synt. Equality-checker									
Metavariable-solver									
WHNormalizer									

Still not feasible to maintain / develop further.

Boilerplate in Menkar

		Π	λ	app	Σ	pair	fst	snd	...
Renaming		deriving Functor **(BAD PERFORMANCE)**							
Substitution		deriving Monad generically **(BAD PERFORMANCE)**							
Scope-checker									
Pretty-printer									
Type-checker	Analyzer								
Def. Equality-checker									
Synt. Equality-checker									
Metavariable-solver									
WHNormalizer									

Still not feasible to maintain / develop further.

Boilerplate in Menkar

		Π	λ	app	Σ	pair	fst	snd	...
		Algebraic operation							
Renaming	Model/ Algebra	Fold							
Substitution									
Scope-checker									
Pretty-printer									
Type-checker									
Def. Equality-checker									
Synt. Equality-checker									
Metavariable-solver									
WHNormalizer									

Still not feasible to maintain / develop further.

Solution: Generic Programming

AACMM21: Allais, Atkey, Chapman, McBride & McKinna, 2021: A Type- and Scope-Safe Universe of Syntaxes with Binding

- Universe of syntaxes
- Semantics of a syntax:

$$\frac{t \in \mathbf{Syntax}(\Delta \vdash T) \qquad env \in \mathscr{V}(\Gamma \vdash \Delta)}{[\![t]\!]_{env} \in \mathscr{C}(\Gamma \vdash T)}$$

- Fully formalized in Agda
- Many applications, implemented in Agda

FS22: Fiore & Szamozvancev, 2022: Formal Metatheory of Second Order Abstract Syntax

- Eq-free SOMATs (second order multisorted algebraic theories)
- Semantics: Models in algebraic sense
- Fully formalized in Agda
- Strictly more general
- Mathematically more elegant/profound

Solution: Generic Programming

AACMM21: Allais, Atkey, Chapman, McBride & McKinna, 2021: A Type- and Scope-Safe Universe of Syntaxes with Binding

- Universe of syntaxes
- Semantics of a syntax:

$$\frac{t \in \mathbf{Syntax}(\Delta \vdash T) \qquad env \in \mathscr{V}(\Gamma \vdash \Delta)}{[\![t]\!]_{env} \in \mathscr{C}(\Gamma \vdash T)}$$

- Fully formalized in Agda
- Many applications, implemented in Agda

FS22: Fiore & Szamozvancev, 2022: Formal Metatheory of Second Order Abstract Syntax

- Eq-free SOMATs (second order multisorted algebraic theories)
- Semantics: Models in algebraic sense
- Fully formalized in Agda
- Strictly more general
- Mathematically more elegant/profound

Solution: Generic Programming

AACMM21: Allais, Atkey, Chapman, McBride & McKinna, 2021: A Type- and Scope-Safe Universe of Syntaxes with Binding

- Universe of syntaxes
- Semantics of a syntax:

$$\frac{t \in \mathbf{Syntax}(\Delta \vdash T) \qquad env \in \mathscr{V}(\Gamma \vdash \Delta)}{[\![t]\!]_{env} \in \mathscr{C}(\Gamma \vdash T)}$$

- Fully formalized in Agda
- Many applications, implemented in Agda

FS22: Fiore & Szamozvancev, 2022: Formal Metatheory of Second Order Abstract Syntax

- Eq-free SOMATs (second order multisorted algebraic theories)
- Semantics: Models in algebraic sense
- Fully formalized in Agda
- Strictly more general
- Mathematically more elegant/profound

Solution: Generic Programming

AACMM21: Allais, Atkey, Chapman, McBride & McKinna, 2021: A Type- and Scope-Safe Universe of Syntaxes with Binding

- Universe of syntaxes
- Semantics of a syntax:

$$\frac{t \in \mathbf{Syntax}(\Delta \vdash T) \qquad env \in \mathscr{V}(\Gamma \vdash \Delta)}{[\![t]\!]_{env} \in \mathscr{C}(\Gamma \vdash T)}$$

- Fully formalized in Agda
- Many applications, implemented in Agda

FS22: Fiore & Szamozvancev, 2022: Formal Metatheory of Second Order Abstract Syntax

- Eq-free SOMATs (second order multisorted algebraic theories)
- Semantics: Models in algebraic sense
- Fully formalized in Agda
- Strictly more general
- Mathematically more elegant/profound

Solution: Generic Programming

AACMM21: Allais, Atkey, Chapman, McBride & McKinna, 2021: A Type- and Scope-Safe Universe of Syntaxes with Binding

- Universe of syntaxes
- Semantics of a syntax:

$$\frac{t \in \mathbf{Syntax}(\Delta \vdash T) \qquad env \in \mathscr{V}(\Gamma \vdash \Delta)}{[\![t]\!]_{env} \in \mathscr{C}(\Gamma \vdash T)}$$

- Fully formalized in Agda
- Many applications, implemented in Agda

FS22: Fiore & Szamozvancev, 2022: Formal Metatheory of Second Order Abstract Syntax

- Eq-free SOMATs (second order multisorted algebraic theories)
- Semantics: Models in algebraic sense
- Fully formalized in Agda
- Strictly more general
- Mathematically more elegant/profound

Solution: Generic Programming

AACMM21: Allais, Atkey, Chapman, McBride & McKinna, 2021: A Type- and Scope-Safe Universe of Syntaxes with Binding

- Universe of syntaxes
- Semantics of a syntax:

$$\frac{t \in \textbf{Syntax}(\Delta \vdash T) \qquad env \in \mathscr{V}(\Gamma \vdash \Delta)}{[\![t]\!]_{env} \in \mathscr{C}(\Gamma \vdash T)}$$

- Fully formalized in Agda
- Many applications, implemented in Agda

FS22: Fiore & Szamozvancev, 2022: Formal Metatheory of Second Order Abstract Syntax

- Eq-free SOMATs (second order multisorted algebraic theories)
- Semantics: Models in algebraic sense
- Fully formalized in Agda
- Strictly more general
- Mathematically more elegant/profound

Solution: Generic Programming

AACMM21: Allais, Atkey, Chapman, McBride & McKinna, 2021: A Type- and Scope-Safe Universe of Syntaxes with Binding

- Universe of syntaxes
- Semantics of a syntax:

$$\frac{t \in \textbf{Syntax}(\Delta \vdash T) \qquad env \in \mathscr{V}(\Gamma \vdash \Delta)}{[\![t]\!]_{env} \in \mathscr{C}(\Gamma \vdash T)}$$

- Fully formalized in Agda
- Many applications, implemented in Agda

FS22: Fiore & Szamozvancev, 2022: Formal Metatheory of Second Order Abstract Syntax

- Eq-free SOMATs (second order multisorted algebraic theories)
- Semantics: Models in algebraic sense
- Fully formalized in Agda
- Strictly more general
- Mathematically more elegant/profound

Solution: Generic Programming

AACMM21: Allais, Atkey, Chapman, McBride & McKinna, 2021: A Type- and Scope-Safe Universe of Syntaxes with Binding

- Universe of syntaxes
- Semantics of a syntax:

$$\frac{t \in \mathbf{Syntax}(\Delta \vdash T) \qquad env \in \mathscr{V}(\Gamma \vdash \Delta)}{[\![t]\!]_{env} \in \mathscr{C}(\Gamma \vdash T)}$$

- Fully formalized in Agda
- Many applications, implemented in Agda

FS22: Fiore & Szamozvancev, 2022: Formal Metatheory of Second Order Abstract Syntax

- Eq-free SOMATs (second order multisorted algebraic theories)
- Semantics: Models in algebraic sense
- Fully formalized in Agda
- Strictly more general
- Mathematically more elegant/profound

Results of AACMM21

Universe of syntaxes contains:

- Lambda-calculi
  - untyped
  - simply-typed
  - bidir'ed (check/infer)
  - $\rightarrow$ with common extensions
- ...
- All SOMATs.

Results:

- Renaming, once and for all,
- Substitution, once and for all,
- Pretty-printing, maximally generically,
- Unscoped syntax, once and for all,
- Scope-checking, once and for all,
- Bidir. type-checking & elaboration: Bidir. LC $\rightarrow$ STLC, with clear potential for generalization,
- Desugaring **let**, once and for all,
- Unsafe NbE, somewhat generically,
- ...

Results of AACMM21

Universe of syntaxes contains:

- Lambda-calculi
  - untyped
  - simply-typed
  - bidir'ed (check/infer)
  - $\rightarrow$ with common extensions
- ...
- All SOMATs.

Results:

- Renaming, once and for all,
- Substitution, once and for all,
- Pretty-printing, maximally generically,
- Unscoped syntax, once and for all,
- Scope-checking, once and for all,
- Bidir. type-checking & elaboration: Bidir. LC $\rightarrow$ STLC, with clear potential for generalization,
- Desugaring **let**, once and for all,
- Unsafe NbE, somewhat generically,
- ...

Results of AACMM21

Universe of syntaxes contains:

- Lambda-calculi
  - untyped
  - simply-typed
  - bidir'ed (check/infer)
  - $\rightarrow$ with common extensions
- . . .
- All SOMATs.

Results:

- Renaming, once and for all,
- Substitution, once and for all,
- Pretty-printing, maximally generically,
- Unscoped syntax, once and for all,
- Scope-checking, once and for all,
- Bidir. type-checking & elaboration: Bidir. LC $\rightarrow$ STLC, with clear potential for generalization,
- Desugaring **let**, once and for all,
- Unsafe NbE, somewhat generically,
- . . .

Results of AACMM21

Universe of syntaxes contains:

- Lambda-calculi
  - untyped
  - simply-typed
  - bidir'ed (check/infer)
  - $\rightarrow$ with common extensions
- ...
- All SOMATs.

Results:

- Renaming, once and for all,
- Substitution, once and for all,
- Pretty-printing, maximally generically,
- Unscoped syntax, once and for all,
- Scope-checking, once and for all,
- Bidir. type-checking & elaboration: Bidir. LC $\rightarrow$ STLC, with clear potential for generalization,
- Desugaring **let**, once and for all,
- Unsafe NbE, somewhat generically,
- ...

Results of AACMM21

Universe of syntaxes contains:

- Lambda-calculi
  - untyped
  - simply-typed
  - bidir'ed (check/infer)
  - $\rightarrow$ with common extensions
- ...
- All SOMATs.

Results:

- Renaming, once and for all,
- Substitution, once and for all,
- Pretty-printing, maximally generically,
- Unscoped syntax, once and for all,
- Scope-checking, once and for all,
- Bidir. type-checking & elaboration: Bidir. LC $\rightarrow$ STLC, with clear potential for generalization,
- Desugaring **let**, once and for all,
- Unsafe NbE, somewhat generically,
- ...

Results of AACMM21

Universe of syntaxes contains:

- Lambda-calculi
  - untyped
  - simply-typed
  - bidir'ed (check/infer)
  - $\rightarrow$ with common extensions
- ...
- All SOMATs.

Results:

- Renaming, once and for all,
- Substitution, once and for all,
- Pretty-printing, maximally generically,
- Unscoped syntax, once and for all,
- Scope-checking, once and for all,
- Bidir. type-checking & elaboration: Bidir. LC $\rightarrow$ STLC, with clear potential for generalization,
- Desugaring **let**, once and for all,
- Unsafe NbE, somewhat generically,
- ...

Results of AACMM21

Universe of syntaxes contains:

- Lambda-calculi
  - untyped
  - simply-typed
  - bidir'ed (check/infer)
  - $\rightarrow$ with common extensions
- ...
- All SOMATs.

Results:

- Renaming, once and for all,
- Substitution, once and for all,
- Pretty-printing, maximally generically,
- Unscoped syntax, once and for all,
- Scope-checking, once and for all,
- Bidir. type-checking & elaboration: Bidir. LC $\rightarrow$ STLC, with clear potential for generalization,
- Desugaring **let**, once and for all,
- Unsafe NbE, somewhat generically,
- ...

Eq-free Algebraic Theory

(Fin.) Container:

Op : **Set**, $arity : Op \to \mathbb{N}$

Syntax functor F : **Set** $\to$ **Set**:

$F\,X = \Sigma(op : Op).\mathbf{Vec}_{arity\ op}\,X$.

Term monad $F^*\,X \cong X \uplus F\,F^*\,X$.

Model = F-algebra = F^*-monad-algebra.

Algebraic Theory

Equation laws have type

$\Sigma(X : \mathbf{Set}).F^*\,X \times F^*\,X$

Term monad $M\,X = F^*\,X / \sim$.

Model = M-monad-algebra.

Example (**Pointed magma**)

$Op = \{e, *\}$

$arity\ e = 0$, $arity\ * = 2$

$p \in F\,X = e \mid x_1 * x_2$

$t \in F^*\,X = x \mid e \mid t_1 * t_2$

Model = pointed magma.

Example (**Monoid**)

$lunit = (\{x\}, e * x, x)$

$runit = (\{x\}, x * e, x)$

$assoc = (\{x, y, z\}, (x * y) * z, x * (y * z))$

$M \cong$ **List**

Model = monoid.

Eq-free Algebraic Theory

(Fin.) Container:

$Op : \mathbf{Set}$, $\quad$ $arity : Op \to \mathbb{N}$

Syntax functor $F : \mathbf{Set} \to \mathbf{Set}$:

$F\,X = \Sigma(op : Op).\mathbf{Vec}_{arity\ op}\,X$.

Term monad $F^*\,X \cong X \uplus F\,F^*\,X$.

Model = F-algebra = F^*-monad-algebra.

Algebraic Theory

Equation laws have type

$\Sigma(X : \mathbf{Set}).F^*\,X \times F^*\,X$

Term monad $M\,X = F^*\,X / \sim$.

Model = M-monad-algebra.

Example (**Pointed magma**)

$Op = \{e, *\}$

$arity\ e = 0, \quad arity * = 2$

$p \in F\,X = e \mid x_1 * x_2$

$t \in F^*\,X = x \mid e \mid t_1 * t_2$

Model = pointed magma.

Example (**Monoid**)

$lunit = (\{x\}, e * x, x)$

$runit = (\{x\}, x * e, x)$

$assoc = (\{x, y, z\}, (x * y) * z, x * (y * z))$

$M \cong \mathbf{List}$

Model = monoid.

Eq-free Algebraic Theory

(Fin.) Container:

$Op : \mathbf{Set}, \qquad arity : Op \to \mathbb{N}$

Syntax functor $F : \mathbf{Set} \to \mathbf{Set}$:

$F\,X = \Sigma(op : Op).\mathbf{Vec}_{arity\ op}\,X$.

Term monad $F^*\,X \cong X \uplus F\,F^*\,X$.

Model = F-algebra = F^*-monad-algebra.

Algebraic Theory

Equation laws have type

$\Sigma(X : \mathbf{Set}).F^*\,X \times F^*\,X$

Term monad $M\,X = F^*\,X/\sim$.

Model = M-monad-algebra.

Example (**Pointed magma**)

$Op = \{e, *\}$

$arity\ e = 0, \quad arity * = 2$

$p \in F\,X = e \mid x_1 * x_2$

$t \in F^*\,X = x \mid e \mid t_1 * t_2$

Model = pointed magma.

Example (**Monoid**)

$lunit = (\{x\}, e * x, x)$

$runit = (\{x\}, x * e, x)$

$assoc = (\{x, y, z\}, (x * y) * z, x * (y * z))$

$M \cong \mathbf{List}$

Model = monoid.

Eq-free Algebraic Theory

(Fin.) Container:

Op : **Set**, $\quad arity : Op \to \mathbb{N}$

Syntax functor F : **Set** $\to$ **Set**:

$F\,X = \Sigma(op : Op).\mathbf{Vec}_{arity\ op}\,X.$

Term monad $F^*\,X \cong X \uplus F\,F^*\,X.$

Model = F-algebra = F^*-monad-algebra.

Algebraic Theory

Equation laws have type

$\Sigma(X : \mathbf{Set}).F^*\,X \times F^*\,X$

Term monad $M\,X = F^*\,X/\sim.$

Model = M-monad-algebra.

Example (**Pointed magma**)

$Op = \{e, *\}$

$arity\ e = 0, \quad arity\ * = 2$

$p \in F\,X = e \mid x_1 * x_2$

$t \in F^*\,X = x \mid e \mid t_1 * t_2$

Model = pointed magma.

Example (**Monoid**)

$lunit = (\{x\}, e * x, x)$

$runit = (\{x\}, x * e, x)$

$assoc = (\{x, y, z\}, (x * y) * z, x * (y * z))$

$M \cong$ **List**

Model = monoid.

Eq-free Algebraic Theory

(Fin.) Container:

Op : **Set**, $\quad$ $arity : Op \to \mathbb{N}$

Syntax functor F : **Set** $\to$ **Set**:

$F\,X = \Sigma(op : Op).\mathbf{Vec}_{arity\ op}\,X.$

Term monad $F^*\,X \cong X \uplus F\,F^*\,X.$

Model = F-algebra = F^*-monad-algebra.

Algebraic Theory

Equation laws have type

$\Sigma(X : \mathbf{Set}).F^*\,X \times F^*\,X$

Term monad $M\,X = F^*\,X / \sim.$

Model = M-monad-algebra.

Example (**Pointed magma**)

$Op = \{e, *\}$

$arity\ e = 0, \quad arity\ * = 2$

$p \in F\,X = e \mid x_1 * x_2$

$t \in F^*\,X = x \mid e \mid t_1 * t_2$

Model = pointed magma.

Example (**Monoid**)

$lunit = (\{x\}, e * x, x)$

$runit = (\{x\}, x * e, x)$

$assoc = (\{x, y, z\}, (x * y) * z, x * (y * z))$

$M \cong \mathbf{List}$

Model = monoid.

Eq-free Algebraic Theory

(Fin.) Container:
Op : **Set**, $\quad$ $arity : Op \to \mathbb{N}$

Syntax functor F : **Set** $\to$ **Set**:
$F\,X = \Sigma(op : Op).\mathbf{Vec}_{arity\ op}\,X$.

Term monad $F^*\,X \cong X \uplus F\,F^*\,X$.

Model = F-algebra = F^*-monad-algebra.

Algebraic Theory

Equation laws have type
$\Sigma(X : \mathbf{Set}).F^*\,X \times F^*\,X$

Term monad $M\,X = F^*\,X/\sim$.

Model = M-monad-algebra.

Example (**Pointed magma**)

$Op = \{e, *\}$
$arity\ e = 0, \quad arity\ * = 2$

$p \in F\,X = e \mid x_1 * x_2$

$t \in F^*\,X = x \mid e \mid t_1 * t_2$

Model = pointed magma.

Example (**Monoid**)

$lunit = (\{x\}, e * x, x)$
$runit = (\{x\}, x * e, x)$
$assoc = (\{x, y, z\}, (x * y) * z, x * (y * z))$

$M \cong$ **List**

Model = monoid.

Eq-free Algebraic Theory

(Fin.) Container:

Op : **Set**, $\quad$ $arity : Op \to \mathbb{N}$

Syntax functor $F : \mathbf{Set} \to \mathbf{Set}$:

$F\,X = \Sigma(op : Op).\mathbf{Vec}_{arity\ op}\,X$.

Term monad $F^*\,X \cong X \uplus F\,F^*\,X$.

Model = F-algebra = F^*-monad-algebra.

Algebraic Theory

Equation laws have type

$\Sigma(X : \mathbf{Set}).F^*\,X \times F^*\,X$

Term monad $M\,X = F^*\,X/\sim$.

Model = M-monad-algebra.

Example (**Pointed magma**)

$Op = \{e, *\}$

$arity\ e = 0, \quad arity\ * = 2$

$p \in F\,X = e \mid x_1 * x_2$

$t \in F^*\,X = x \mid e \mid t_1 * t_2$

Model = pointed magma.

Example (**Monoid**)

$lunit = (\{x\}, e * x, x)$

$runit = (\{x\}, x * e, x)$

$assoc = (\{x, y, z\}, (x * y) * z, x * (y * z))$

$M \cong \mathbf{List}$

Model = monoid.

Eq-free Algebraic Theory

(Fin.) Container:

$Op : \mathbf{Set}, \quad arity : Op \to \mathbb{N}$

Syntax functor $F : \mathbf{Set} \to \mathbf{Set}$:

$F\, X = \Sigma(op : Op).\mathbf{Vec}_{arity\ op}\, X.$

Term monad $F^*\, X \cong X \uplus F\, F^*\, X$.

Model = F-algebra = F^*-monad-algebra.

Multisorted Algebraic Theories (MATs)

Eq-free Multisorted Algebraic Theory

(Fin.) Container:

$Op : \mathbf{Set}, \qquad arity : Op \to \mathbb{N}$

Syntax functor $F : \mathbf{Set} \to \mathbf{Set}$:

$F\,X = \Sigma(op : Op).\mathbf{Vec}_{arity\;op}\,X.$

Term monad $F^*\,X \cong X \uplus F\,F^*\,X.$

Model = F-algebra = F^*-monad-algebra.

Eq-free Multisorted Algebraic Theory

Fix $Sort$: **Set**

(Fin.) Container:
Op : **Set**, $arity : Op \to \mathbb{N}$

Syntax functor F : **Set** $\to$ **Set**:
$F\,X = \Sigma(op : Op).\mathbf{Vec}_{arity\ op}\,X.$

Term monad $F^*\,X \cong X \uplus F\,F^*\,X.$

Model = F-algebra = F^*-monad-algebra.

Multisorted Algebraic Theories (MATs)

Eq-free Multisorted Algebraic Theory

Fix $Sort$: **Set**

(Fin.) Container:

$Op : \mathbf{Set}, \qquad arity : Op \to \mathbb{N}$

Syntax functor $F : \mathbf{Set} \to \mathbf{Set}$:

$F\,X = \Sigma(op : Op).\mathbf{Vec}_{arity\ op}\,X$.

Term monad $F^*\,X \cong X \uplus F\,F^*\,X$.

Model = F-algebra = F^*-monad-algebra.

Example (Typed boolean arithmetic)

$Sort = \{bool, base\}$

$tt, ff : Op\,bool$

$if : \forall\{\mathrm{S}\}.Op\,\mathrm{S}$

$arity\ tt = []$

$arity\ ff = []$

$arity\ (if\{\mathrm{S}\}) = [bool, \mathrm{S}, \mathrm{S}]$

Syntax functor F:

$$\frac{}{tt, ff \in F\,X\,bool} \qquad \frac{b \in X\,bool \quad x, y \in X\,\mathrm{S}}{if(b, x, y) \in F\,X\,\mathrm{S}}$$

$$\frac{x \in X\,\mathrm{S}}{x \in F^*\,X\,\mathrm{S}}$$

Eq-free Multisorted Algebraic Theory

Fix $Sort$: **Set**

(Fin.) Indexed Container:

$Op : Sort \to \mathbf{Set}$, $\quad arity : Op \to \mathbb{N}$

Syntax functor $F : \mathbf{Set} \to \mathbf{Set}$:

$F\,X = \Sigma(op : Op).\mathbf{Vec}_{arity\;op}\,X$.

Term monad $F^*\,X \cong X \uplus F\,F^*\,X$.

Model = F-algebra = F^*-monad-algebra.

Example (Typed boolean arithmetic)

$Sort = \{bool, base\}$

$tt, ff : Op\,bool$

$if : \forall\{\mathrm{S}\}.Op\,\mathrm{S}$

$arity\;tt = []$

$arity\;ff = []$

$arity\;(if\{\mathrm{S}\}) = [bool, \mathrm{S}, \mathrm{S}]$

Syntax functor F:

$$\frac{}{tt, ff \in F\,X\,bool} \qquad \frac{b \in X\,bool \quad x, y \in X\,\mathrm{S}}{if(b, x, y) \in F\,X\,\mathrm{S}}$$

$$\frac{x \in X\,\mathrm{S}}{x \in F^*\,X\,\mathrm{S}}$$

Eq-free Multisorted Algebraic Theory

Fix $Sort$: **Set**

(Fin.) Indexed Container:

$Op : Sort \to \mathbf{Set}$

$arity : \forall\{S\}.Op\,S \to \mathbf{List}\,Sort$

Syntax functor $F : \mathbf{Set} \to \mathbf{Set}$:

$F\,X = \Sigma(op : Op).\mathbf{Vec}_{arity\;op}\,X.$

Term monad $F^*\,X \cong X \uplus F\,F^*\,X.$

Model = F-algebra = F^*-monad-algebra.

Example (Typed boolean arithmetic)

$Sort = \{bool, base\}$

$tt, ff : Op\,bool$

$if : \forall\{S\}.Op\,S$

$arity\;tt = []$

$arity\;ff = []$

$arity\;(if\{S\}) = [bool, S, S]$

Syntax functor F:

$$\frac{}{tt, ff \in F\,X\,bool} \qquad \frac{b \in X\,bool \quad x, y \in X\,S}{if(b, x, y) \in F\,X\,S}$$

$$\frac{x \in X\,S}{x \in F^*\,X\,S}$$

Eq-free Multisorted Algebraic Theory

Fix $Sort$: **Set**

(Fin.) Indexed Container:

$Op : Sort \to \mathbf{Set}$

$arity : \forall\{S\}.Op\,S \to \mathbf{List}\,Sort$

Syntax functor F : **Set** $\to$ **Set**:

$F\,X = \Sigma(op : Op).\mathbf{Vec}_{arity\ op}\,X$.

Term monad $F^*\,X \cong X \uplus F\,F^*\,X$.

Model = F-algebra = F^*-monad-algebra.

Example (Typed boolean arithmetic)

$Sort = \{bool, base\}$

$tt, ff : Op\,bool$

$if : \forall\{S\}.Op\,S$

$arity\ tt = []$

$arity\ ff = []$

$arity\ (if\{S\}) = [bool, S, S]$

Syntax functor F:

$$\frac{}{tt, ff \in F\,X\,bool} \qquad \frac{b \in X\,bool \quad x, y \in X\,S}{if(b, x, y) \in F\,X\,S}$$

$$\frac{x \in X\,S}{x \in F^*\,X\,S}$$

Eq-free Multisorted Algebraic Theory

Fix $Sort : \mathbf{Set}$

(Fin.) Indexed Container:

$Op : Sort \to \mathbf{Set}$

$arity : \forall\{S\}.Op\,S \to \mathbf{List}\,Sort$

Syntax functor $F : \mathbf{Set}^{Sort} \to \mathbf{Set}^{Sort}$:

$F\,X = \Sigma(op : Op).\mathbf{Vec}_{arity\ op}\,X.$

Term monad $F^*\,X \cong X \uplus F\,F^*\,X.$

Model = F-algebra = F^*-monad-algebra.

Example (Typed boolean arithmetic)

$Sort = \{bool, base\}$

$tt, ff : Op\,bool$

$if : \forall\{S\}.Op\,S$

$arity\ tt = []$

$arity\ ff = []$

$arity\,(if\{S\}) = [bool, S, S]$

Syntax functor F:

$$\frac{}{tt, ff \in F\,X\,bool} \qquad \frac{b \in X\,bool \quad x, y \in X\,S}{if(b, x, y) \in F\,X\,S}$$

$$\frac{x \in X\,S}{x \in F^*\,X\,S}$$

Multisorted Algebraic Theories (MATs)

Eq-free Multisorted Algebraic Theory

Fix $Sort : \mathbf{Set}$

(Fin.) Indexed Container:

$Op : Sort \to \mathbf{Set}$

$arity : \forall\{S\}.Op\,S \to \mathbf{List}\,Sort$

Syntax functor $F : \mathbf{Set}^{Sort} \to \mathbf{Set}^{Sort}$:

$F\,X\,S = \Sigma(op : Op\,S).\mathbf{ListP}_{arity\ op}\,X.$

Term monad $F^*\,X \cong X \uplus F\,F^*\,X.$

Model = F-algebra = F^*-monad-algebra.

Example (Typed boolean arithmetic)

$Sort = \{bool, base\}$

$tt, ff : Op\,bool$

$if : \forall\{S\}.Op\,S$

$arity\ tt = []$

$arity\ ff = []$

$arity\ (if\{S\}) = [bool, S, S]$

Syntax functor F:

$$\frac{}{tt, ff \in F\,X\,bool} \qquad \frac{b \in X\,bool \quad x, y \in X\,S}{if(b, x, y) \in F\,X\,S}$$

$$\frac{x \in X\,S}{x \in F^*\,X\,S}$$

Multisorted Algebraic Theories (MATs)

Eq-free Multisorted Algebraic Theory

Fix $Sort : \mathbf{Set}$

(Fin.) Indexed Container:

$Op : Sort \to \mathbf{Set}$

$arity : \forall\{S\}.Op\,S \to \mathbf{List}\,Sort$

Syntax functor $F : \mathbf{Set}^{Sort} \to \mathbf{Set}^{Sort}$:

$F\,X\,S = \Sigma(op : Op\,S).\mathbf{ListP}_{arity\ op}\,X.$

Term monad $F^*\,X \cong X \uplus F\,F^*\,X$.

Model = F-algebra = F^*-monad-algebra.

Example (Typed boolean arithmetic)

$Sort = \{bool, base\}$

$tt, ff : Op\,bool$

$if : \forall\{S\}.Op\,S$

$arity\ tt = []$

$arity\ ff = []$

$arity\ (if\{S\}) = [bool, S, S]$

Syntax functor F:

$$\frac{}{tt, ff \in F\,X\,bool} \qquad \frac{b \in X\,bool \quad x, y \in X\,S}{if(b, x, y) \in F\,X\,S}$$

$$\frac{x \in X\,S}{x \in F^*\,X\,S}$$

Multisorted Algebraic Theories (MATs)

Eq-free Multisorted Algebraic Theory

Fix $\mathit{Sort} : \mathbf{Set}$

(Fin.) Indexed Container:
$\mathit{Op} : \mathit{Sort} \to \mathbf{Set}$
$\mathit{arity} : \forall\{\mathrm{S}\}.\mathit{Op}\,\mathrm{S} \to \mathbf{List}\,\mathit{Sort}$

Syntax functor $F : \mathbf{Set}^{\mathit{Sort}} \to \mathbf{Set}^{\mathit{Sort}}$:
$F\,X\,\mathrm{S} = \Sigma(\mathit{op} : \mathit{Op}\,\mathrm{S}).\mathbf{ListP}_{\mathit{arity}\ \mathit{op}}\,X.$

Term monad $F^*\,X\,\mathrm{S} \cong X\,\mathrm{S} \uplus F\,(F^*\,X)\,\mathrm{S}.$

Model = F-algebra = F^*-monad-algebra.

Example (Typed boolean arithmetic)

$\mathit{Sort} = \{\mathit{bool}, \mathit{base}\}$

$\mathit{tt}, \mathit{ff} : \mathit{Op}\,\mathit{bool}$
$\mathit{if} : \forall\{\mathrm{S}\}.\mathit{Op}\,\mathrm{S}$

$\mathit{arity}\ \mathit{tt} = []$
$\mathit{arity}\ \mathit{ff} = []$
$\mathit{arity}\ (\mathit{if}\{\mathrm{S}\}) = [\mathit{bool}, \mathrm{S}, \mathrm{S}]$

Syntax functor F:

$$\frac{}{\mathit{tt}, \mathit{ff} \in F\,X\,\mathit{bool}} \qquad \frac{b \in X\,\mathit{bool} \quad x, y \in X\,\mathrm{S}}{\mathit{if}(b, x, y) \in F\,X\,\mathrm{S}}$$

$$\frac{x \in X\,\mathrm{S}}{x \in F^*\,X\,\mathrm{S}}$$

Eq-free Multisorted Algebraic Theory

Fix $\mathit{Sort} : \mathbf{Set}$

(Fin.) Indexed Container:

$\mathit{Op} : \mathit{Sort} \to \mathbf{Set}$

$\mathit{arity} : \forall\{\mathrm{S}\}.\mathit{Op}\,\mathrm{S} \to \mathbf{List}\,\mathit{Sort}$

Syntax functor $F : \mathbf{Set}^{\mathit{Sort}} \to \mathbf{Set}^{\mathit{Sort}}$:

$F\,X\,\mathrm{S} = \Sigma(\mathit{op} : \mathit{Op}\,\mathrm{S}).\mathbf{ListP}_{\mathit{arity}\,\mathit{op}}\,X.$

Term monad $F^*\,X\,\mathrm{S} \cong X\,\mathrm{S} \uplus F\,(F^*\,X)\,\mathrm{S}.$

Model = F-algebra = F^*-monad-algebra.

Example (Typed boolean arithmetic)

$\mathit{Sort} = \{\mathit{bool}, \mathit{base}\}$

$\mathit{tt}, \mathit{ff} : \mathit{Op}\,\mathit{bool}$

$\mathit{if} : \forall\{\mathrm{S}\}.\mathit{Op}\,\mathrm{S}$

$\mathit{arity}\,\mathit{tt} = []$

$\mathit{arity}\,\mathit{ff} = []$

$\mathit{arity}\,(\mathit{if}\{\mathrm{S}\}) = [\mathit{bool}, \mathrm{S}, \mathrm{S}]$

Free monad F^*:

$$\frac{}{\mathit{tt}, \mathit{ff} \in F^*\,X\,\mathit{bool}} \qquad \frac{\mathit{tb} \in F^*\,X\,\mathit{bool} \quad \mathit{tx}, \mathit{ty} \in F^*\,X\,\mathrm{S}}{\mathit{if}(\mathit{tb}, \mathit{tx}, \mathit{ty}) \in F^*\,X\,\mathrm{S}}$$

$$\frac{x \in X\,\mathrm{S}}{x \in F^*\,X\,\mathrm{S}}$$

Multisorted Algebraic Theories (MATs)

Eq-free Multisorted Algebraic Theory

Fix $Sort : \mathbf{Set}$

(Fin.) Indexed Container:
$Op : Sort \to \mathbf{Set}$
$arity : \forall\{\mathrm{S}\}.Op\,\mathrm{S} \to \mathbf{List}\,Sort$

Syntax functor $F : \mathbf{Set}^{Sort} \to \mathbf{Set}^{Sort}$:
$F\,X\,\mathrm{S} = \Sigma(op : Op\,\mathrm{S}).\mathbf{ListP}_{arity\ op}\,X.$

Term monad $F^*\,X\,\mathrm{S} \cong X\,\mathrm{S} \uplus F\,(F^*\,X)\,\mathrm{S}.$

Model = F-algebra = F^*-monad-algebra.

Example (Typed boolean arithmetic)

$Sort = \{bool, base\}$

$tt, ff : Op\,bool$
$if : \forall\{\mathrm{S}\}.Op\,\mathrm{S}$

$arity\ tt = []$
$arity\ ff = []$
$arity\ (if\{\mathrm{S}\}) = [bool, \mathrm{S}, \mathrm{S}]$

Free monad F^*:

$$\frac{}{tt, ff \in F^*\,X\,bool} \qquad \frac{tb \in F^*\,X\,bool \quad tx, ty \in F^*\,X\,\mathrm{S}}{if(tb, tx, ty) \in F^*\,X\,\mathrm{S}}$$

$$\frac{x \in X\,\mathrm{S}}{x \in F^*\,X\,\mathrm{S}}$$

Eq-free Multisorted Algebraic Theory

Fix $Sort : \mathbf{Set}$

(Fin.) Indexed Container:
$Op : Sort \to \mathbf{Set}$
$arity : \forall\{S\}.Op\, S \to \mathbf{List}\, Sort$

Syntax functor $F : \mathbf{Set}^{Sort} \to \mathbf{Set}^{Sort}$:
$F\, X\, S = \Sigma(op : Op\, S).\mathbf{ListP}_{arity\ op}\, X.$

Term monad $F^*\, X\, S \cong X\, S \uplus F\,(F^*\, X)\, S.$

Model = F-algebra = F^*-monad-algebra.

Eq-free SOMAT

Fix $Sort : \mathbf{Set}$

(Fin.) Indexed Container:
$Op : Sort \to \mathbf{Set}$
$arity : \forall\{S\}.Op\, S \to \mathbf{List}\, Sort$

Syntax functor $F : \mathbf{Set}^{Sort} \to \mathbf{Set}^{Sort}$:
$F\, X\, S = \Sigma(op : Op\, S).\mathbf{ListP}_{arity\; op}\, X.$

Term monad $F^*\, X\, S \cong X\, S \uplus F\,(F^*\, X)\, S.$

Model = F-algebra = F^*-monad-algebra.

Eq-free SOMAT

Fix $Sort$: **Set**

$Ctx := Tele :=$ **List** $Sort$,

$Jud := Ctx \times Sort$.

(Fin.) Indexed Container:

$Op : Sort \to$ **Set**

$arity : \forall\{S\}.Op\,S \to$ **List** $Sort$

Syntax functor F : $\mathbf{Set}^{Sort} \to \mathbf{Set}^{Sort}$:

$F\,X\,S = \Sigma(op : Op\,S).\mathbf{ListP}_{arity\ op}\,X$.

Term monad $F^*\,X\,S \cong X\,S \uplus F\,(F^*\,X)\,S$.

Model = F-algebra = F^*-monad-algebra.

Eq-free SOMAT

Fix $\mathit{Sort} : \mathbf{Set}$

$\mathit{Ctx} := \mathit{Tele} := \mathbf{List}\,\mathit{Sort}$,

$\mathit{Jud} := \mathit{Ctx} \times \mathit{Sort}$.

$\mathit{Op} : \mathit{Sort} \to \mathbf{Set}$

$\mathit{arity} : \forall\{\mathrm{S}\}.\mathit{Op}\,\mathrm{S} \to \mathbf{List}\,(\mathit{Tele} \times \mathit{Sort})$

Syntax functor $F : \mathbf{Set}^{\mathit{Sort}} \to \mathbf{Set}^{\mathit{Sort}}$:

$F\,X\,\mathrm{S} = \Sigma(\mathit{op} : \mathit{Op}\,\mathrm{S}).\mathbf{ListP}_{\mathit{arity}\,\mathit{op}}\,X$.

Term monad $F^*\,X\,\mathrm{S} \cong X\,\mathrm{S} \uplus F\,(F^*\,X)\,\mathrm{S}$.

Model = F-algebra = F^*-monad-algebra.

Second Order Multisorted Algebraic Theories (SOMATs)

Eq-free SOMAT

Fix *Sort* : **Set**

$\mathit{Ctx} := \mathit{Tele} := \mathbf{List}\,\mathit{Sort}$,
$\mathit{Jud} := \mathit{Ctx} \times \mathit{Sort}$.

$\mathit{Op} : \mathit{Sort} \to \mathbf{Set}$
$\mathit{arity} : \forall\{S\}.\mathit{Op}\,S \to \mathbf{List}\,(\mathit{Tele} \times \mathit{Sort})$

Syntax functor $F : \mathbf{Set}^{\mathit{Sort}} \to \mathbf{Set}^{\mathit{Sort}}$:
$F\,X\,S = \Sigma(\mathit{op} : \mathit{Op}\,S).\mathbf{ListP}_{\mathit{arity}\ \mathit{op}}\,X$.

Term monad $F^*\,X\,S \cong X\,S \uplus F\,(F^*\,X)\,S$.

Model = F-algebra = F^*-monad-algebra.

Example (STLC)

$S \in \mathit{Sort} = \mathit{base} \mid S_1 \Rightarrow S_2$

$\lambda : \forall\{S,T\}.\mathit{Op}\,(S \Rightarrow T)$
$\$: \forall\{S,T\}.\mathit{Op}\,T$

$\mathit{arity}\,\lambda\{S,T\} = [([S],T)]$
$\mathit{arity}\,\$\{S,T\} = [([],S \Rightarrow T),([],S)]$

Syntax functor F:

$$\frac{\mathit{body} \in X(\Gamma + [S] \vdash T)}{\lambda\,\mathit{body} \in F\,X(\Gamma \vdash S \Rightarrow T)}$$

$$\frac{f \in X\,(\Gamma + [] \vdash S \Rightarrow T) \quad x \in X\,(\Gamma + [] \vdash S)}{f\,\$\,x \in F\,X\,(\Gamma \vdash T)} \qquad \frac{x \in X\,\mathit{jud}}{x \in F^*\,X\,\mathit{jud}}$$

Second Order Multisorted Algebraic Theories (SOMATs)

Eq-free SOMAT

Fix $\mathit{Sort} : \mathbf{Set}$

$\mathit{Ctx} := \mathit{Tele} := \mathbf{List}\,\mathit{Sort}$,

$\mathit{Jud} := \mathit{Ctx} \times \mathit{Sort}$.

$\mathit{Op} : \mathit{Sort} \to \mathbf{Set}$

$\mathit{arity} : \forall\{\mathrm{S}\}.\mathit{Op}\,\mathrm{S} \to \mathbf{List}\,(\mathit{Tele} \times \mathit{Sort})$

Syntax functor $F : \mathbf{Set}^{\mathit{Sort}} \to \mathbf{Set}^{\mathit{Sort}}$:

$F\,X\,\mathrm{S} = \Sigma(\mathit{op} : \mathit{Op}\,\mathrm{S}).\mathbf{ListP}_{\mathit{arity}\ \mathit{op}}\,X$.

Term monad $F^*\,X\,\mathrm{S} \cong X\,\mathrm{S} \uplus F\,(F^*\,X)\,\mathrm{S}$.

Model = F-algebra = F^*-monad-algebra.

Example (STLC)

$\mathrm{S} \in \mathit{Sort} = \mathit{base} \mid \mathrm{S}_1 \Rightarrow \mathrm{S}_2$

$\lambda : \forall\{\mathrm{S},\mathrm{T}\}.\mathit{Op}\,(\mathrm{S} \Rightarrow \mathrm{T})$

$\$: \forall\{\mathrm{S},\mathrm{T}\}.\mathit{Op}\,\mathrm{T}$

$\mathit{arity}\,\lambda\{\mathrm{S},\mathrm{T}\} = [([\mathrm{S}],\mathrm{T})]$

$\mathit{arity}\,\$\{\mathrm{S},\mathrm{T}\} = [([],\mathrm{S} \Rightarrow \mathrm{T}),([],\mathrm{S})]$

Syntax functor F:

$$\frac{\mathit{body} \in X(\Gamma \dotplus [\mathrm{S}] \vdash \mathrm{T})}{\lambda\,\mathit{body} \in F\,X(\Gamma \vdash \mathrm{S} \Rightarrow \mathrm{T})}$$

$$\frac{f \in X(\Gamma \dotplus [] \vdash \mathrm{S} \Rightarrow \mathrm{T}) \quad x \in X(\Gamma \dotplus [] \vdash \mathrm{S})}{f\,\$\,x \in F\,X(\Gamma \vdash \mathrm{T})} \qquad \frac{x \in X\,\mathit{jud}}{x \in F^*\,X\,\mathit{jud}}$$

Second Order Multisorted Algebraic Theories (SOMATs)

Eq-free SOMAT

Fix $\mathit{Sort} : \mathbf{Set}$

$\mathit{Ctx} := \mathit{Tele} := \mathbf{List}\,\mathit{Sort}$,

$\mathit{Jud} := \mathit{Ctx} \times \mathit{Sort}$.

$\mathit{Op} : \mathit{Sort} \to \mathbf{Set}$

$\mathit{arity} : \forall\{\mathrm{S}\}.\mathit{Op}\,\mathrm{S} \to \mathbf{List}\,(\mathit{Tele} \times \mathit{Sort})$

Syntax functor $F : \mathbf{Set}^{\mathit{Jud}} \to \mathbf{Set}^{\mathit{Jud}}$:

$F\,X\,\mathrm{S} = \Sigma(\mathit{op} : \mathit{Op}\,\mathrm{S}).\mathbf{ListP}_{\mathit{arity}\ \mathit{op}}\,X$.

Term monad $F^*\,X\,\mathrm{S} \cong X\,\mathrm{S} \uplus F\,(F^*\,X)\,\mathrm{S}$.

Model = F-algebra = F^*-monad-algebra.

Example (STLC)

$\mathrm{S} \in \mathit{Sort} = \mathit{base} \mid \mathrm{S}_1 \Rightarrow \mathrm{S}_2$

$\lambda : \forall\{\mathrm{S},\mathrm{T}\}.\mathit{Op}\,(\mathrm{S} \Rightarrow \mathrm{T})$

$\$: \forall\{\mathrm{S},\mathrm{T}\}.\mathit{Op}\,\mathrm{T}$

$\mathit{arity}\,\lambda\{\mathrm{S},\mathrm{T}\} = [([\mathrm{S}],\mathrm{T})]$

$\mathit{arity}\,\$\{\mathrm{S},\mathrm{T}\} = [([],\mathrm{S} \Rightarrow \mathrm{T}),([],\mathrm{S})]$

Syntax functor F:

$$\frac{\mathit{body} \in X(\Gamma \dotplus [\mathrm{S}] \vdash \mathrm{T})}{\lambda\,\mathit{body} \in F\,X(\Gamma \vdash \mathrm{S} \Rightarrow \mathrm{T})}$$

$$\frac{f \in X\,(\Gamma \dotplus [] \vdash \mathrm{S} \Rightarrow \mathrm{T}) \qquad x \in X\,(\Gamma \dotplus [] \vdash \mathrm{S})}{f\,\$\,x \in F\,X\,(\Gamma \vdash \mathrm{T})} \qquad \frac{x \in X\,\mathit{jud}}{x \in F^*\,X\,\mathit{jud}}$$

Second Order Multisorted Algebraic Theories (SOMATs)

Eq-free SOMAT

Fix $\mathit{Sort} : \mathbf{Set}$

$\mathit{Ctx} := \mathit{Tele} := \mathbf{List}\,\mathit{Sort}$,
$\mathit{Jud} := \mathit{Ctx} \times \mathit{Sort}$.

$\mathit{Op} : \mathit{Sort} \to \mathbf{Set}$
$\mathit{arity} : \forall\{S\}.\mathit{Op}\,S \to \mathbf{List}\,(\mathit{Tele} \times \mathit{Sort})$

Syntax functor $F : \mathbf{Set}^{\mathit{Jud}} \to \mathbf{Set}^{\mathit{Jud}}$:
$F\,X\,(\Gamma \vdash S) = \Sigma(\mathit{op} : \mathit{Op}\,S)$.
$\mathbf{ListP}_{\mathit{arity}\ \mathit{op}}\,(\lambda(\Theta, T).X\,(\Gamma \dotplus \Theta \vdash T))$.

Term monad $F^*\,X\,S \cong X\,S \uplus F\,(F^*\,X)\,S$.

Model = F-algebra = F^*-monad-algebra.

Example (STLC)

$S \in \mathit{Sort} = \mathit{base} \mid S_1 \Rightarrow S_2$

$\lambda : \forall\{S, T\}.\mathit{Op}\,(S \Rightarrow T)$
$\$: \forall\{S, T\}.\mathit{Op}\,T$

$\mathit{arity}\,\lambda\{S, T\} = [([S], T)]$
$\mathit{arity}\,\$\{S, T\} = [([], S \Rightarrow T), ([], S)]$

Syntax functor F:

$$\frac{\mathit{body} \in X(\Gamma \dotplus [S] \vdash T)}{\lambda\,\mathit{body} \in F\,X(\Gamma \vdash S \Rightarrow T)}$$

$$\frac{f \in X(\Gamma \dotplus [] \vdash S \Rightarrow T) \qquad x \in X(\Gamma \dotplus [] \vdash S)}{f\,\$\,x \in F\,X(\Gamma \vdash T)} \qquad \frac{x \in X\,\mathit{jud}}{x \in F^*\,X\,\mathit{jud}}$$

Eq-free SOMAT

Fix $\mathit{Sort} : \mathbf{Set}$

$\mathit{Ctx} := \mathit{Tele} := \mathbf{List}\,\mathit{Sort}$,

$\mathit{Jud} := \mathit{Ctx} \times \mathit{Sort}$.

$\mathit{Op} : \mathit{Sort} \to \mathbf{Set}$

$\mathit{arity} : \forall\{\mathrm{S}\}.\mathit{Op}\,\mathrm{S} \to \mathbf{List}\,(\mathit{Tele} \times \mathit{Sort})$

Syntax functor $F : \mathbf{Set}^{\mathit{Jud}} \to \mathbf{Set}^{\mathit{Jud}}$:

$F\,X\,(\Gamma \vdash \mathrm{S}) = \Sigma(\mathit{op} : \mathit{Op}\,\mathrm{S})$.

$\mathbf{ListP}_{\mathit{arity}\ \mathit{op}}\,(\lambda(\Theta, \mathrm{T}).X\,(\Gamma \dotplus \Theta \vdash \mathrm{T}))$.

Term monad $F^*\,X\,\mathrm{S} \cong X\,\mathrm{S} \uplus F\,(F^*\,X)\,\mathrm{S}$.

Model = F-algebra = F^*-monad-algebra.

Example (STLC)

$\mathrm{S} \in \mathit{Sort} = \mathit{base} \mid \mathrm{S}_1 \Rightarrow \mathrm{S}_2$

$\lambda : \forall\{\mathrm{S}, \mathrm{T}\}.\mathit{Op}\,(\mathrm{S} \Rightarrow \mathrm{T})$

$\$: \forall\{\mathrm{S}, \mathrm{T}\}.\mathit{Op}\,\mathrm{T}$

$\mathit{arity}\,\lambda\{\mathrm{S}, \mathrm{T}\} = [([\mathrm{S}], \mathrm{T})]$

$\mathit{arity}\,\$\{\mathrm{S}, \mathrm{T}\} = [([], \mathrm{S} \Rightarrow \mathrm{T}), ([], \mathrm{S})]$

Syntax functor F:

$$\frac{\mathit{body} \in X(\Gamma \dotplus [\mathrm{S}] \vdash \mathrm{T})}{\lambda\,\mathit{body} \in F\,X(\Gamma \vdash \mathrm{S} \Rightarrow \mathrm{T})}$$

$$\frac{f \in X(\Gamma \dotplus [] \vdash \mathrm{S} \Rightarrow \mathrm{T}) \qquad x \in X(\Gamma \dotplus [] \vdash \mathrm{S})}{f\,\$\,x \in F\,X(\Gamma \vdash \mathrm{T})} \qquad \frac{x \in X\,\mathit{jud}}{x \in F^*\,X\,\mathit{jud}}$$

Second Order Multisorted Algebraic Theories (SOMATs)

Eq-free SOMAT

Fix $Sort$: **Set**

$Ctx := Tele := \mathbf{List}\, Sort$,

$Jud := Ctx \times Sort$.

$Op : Sort \to \mathbf{Set}$

$arity : \forall\{S\}.Op\, S \to \mathbf{List}\,(Tele \times Sort)$

Syntax functor $F : \mathbf{Set}^{Jud} \to \mathbf{Set}^{Jud}$:

$F\, X\,(\Gamma \vdash S) = \Sigma(op : Op\, S).$

$\mathbf{ListP}_{arity\ op}\,(\lambda(\Theta, T).X\,(\Gamma \dotplus \Theta \vdash T)).$

Term monad $F^*\, X\, jud \cong X\, jud \uplus F\,(F^*\, X)\, jud$.

Model = F-algebra = F^*-monad-algebra.

Example (STLC)

$S \in Sort = base \mid S_1 \Rightarrow S_2$

$\lambda : \forall\{S, T\}.Op\,(S \Rightarrow T)$

$\$: \forall\{S, T\}.Op\, T$

$arity\ \lambda\{S, T\} = [([S], T)]$

$arity\ \$\{S, T\} = [([], S \Rightarrow T), ([], S)]$

Syntax functor F:

$$\frac{body \in X(\Gamma \dotplus [S] \vdash T)}{\lambda\ body \in F\, X(\Gamma \vdash S \Rightarrow T)}$$

$$\frac{f \in X(\Gamma \dotplus [] \vdash S \Rightarrow T) \quad x \in X(\Gamma \dotplus [] \vdash S)}{f\,\$\,x \in F\, X(\Gamma \vdash T)} \qquad \frac{x \in X\, jud}{x \in F^*\, X\, jud}$$

Second Order Multisorted Algebraic Theories (SOMATs)

Eq-free SOMAT

Fix $\mathit{Sort} : \mathbf{Set}$

$\mathit{Ctx} := \mathit{Tele} := \mathbf{List}\,\mathit{Sort}$,

$\mathit{Jud} := \mathit{Ctx} \times \mathit{Sort}$.

$\mathit{Op} : \mathit{Sort} \to \mathbf{Set}$

$\mathit{arity} : \forall\{\mathrm{S}\}.\mathit{Op}\,\mathrm{S} \to \mathbf{List}\,(\mathit{Tele} \times \mathit{Sort})$

Syntax functor $F : \mathbf{Set}^{\mathit{Jud}} \to \mathbf{Set}^{\mathit{Jud}}$:

$F\,X\,(\Gamma \vdash \mathrm{S}) = \Sigma(\mathit{op} : \mathit{Op}\,\mathrm{S})$.

$\mathbf{ListP}_{\mathit{arity}\ \mathit{op}}\,(\lambda(\Theta, \mathrm{T}).X\,(\Gamma \dotplus \Theta \vdash \mathrm{T}))$.

Term monad $F^*\,X\,\mathit{jud} \cong X\,\mathit{jud} \uplus F\,(F^*\,X)\,\mathit{jud}$.

Model = F-algebra = F^*-monad-algebra.

Example (STLC)

$\mathrm{S} \in \mathit{Sort} = \mathit{base} \mid \mathrm{S}_1 \Rightarrow \mathrm{S}_2$

$\lambda : \forall\{\mathrm{S}, \mathrm{T}\}.\mathit{Op}\,(\mathrm{S} \Rightarrow \mathrm{T})$

$\$: \forall\{\mathrm{S}, \mathrm{T}\}.\mathit{Op}\,\mathrm{T}$

$\mathit{arity}\ \lambda\{\mathrm{S}, \mathrm{T}\} = [([\mathrm{S}], \mathrm{T})]$

$\mathit{arity}\ \$\{\mathrm{S}, \mathrm{T}\} = [([], \mathrm{S} \Rightarrow \mathrm{T}), ([], \mathrm{S})]$

Syntax functor F:

$$\frac{\mathit{body} \in X(\Gamma \dotplus [\mathrm{S}] \vdash \mathrm{T})}{\lambda\ \mathit{body} \in F\,X(\Gamma \vdash \mathrm{S} \Rightarrow \mathrm{T})}$$

$$\frac{f \in X\,(\Gamma \dotplus [] \vdash \mathrm{S} \Rightarrow \mathrm{T}) \qquad x \in X\,(\Gamma \dotplus [] \vdash \mathrm{S})}{f\,\$\,x \in F\,X\,(\Gamma \vdash \mathrm{T})} \qquad \frac{x \in X\,\mathit{jud}}{x \in F^*\,X\,\mathit{jud}}$$

Second Order Multisorted Algebraic Theories (SOMATs)

Eq-free SOMAT

Fix $\mathit{Sort} : \mathbf{Set}$

$\mathit{Ctx} := \mathit{Tele} := \mathbf{List}\,\mathit{Sort}$,

$\mathit{Jud} := \mathit{Ctx} \times \mathit{Sort}$.

$\mathit{Op} : \mathit{Sort} \to \mathbf{Set}$

$\mathit{arity} : \forall\{\mathrm{S}\}.\mathit{Op}\,\mathrm{S} \to \mathbf{List}\,(\mathit{Tele} \times \mathit{Sort})$

Syntax functor $F : \mathbf{Set}^{\mathit{Jud}} \to \mathbf{Set}^{\mathit{Jud}}$:

$F\,X\,(\Gamma \vdash \mathrm{S}) = \Sigma(\mathit{op} : \mathit{Op}\,\mathrm{S})$.

$\mathbf{ListP}_{\mathit{arity}\,\mathit{op}}\,(\lambda(\Theta, \mathrm{T}).X\,(\Gamma \dotplus \Theta \vdash \mathrm{T}))$.

Term monad $F^*\,X\,\mathit{jud} \cong X\,\mathit{jud} \uplus F\,(F^*\,X)\,\mathit{jud}$.

Model = F-algebra = F^*-monad-algebra.

Example (STLC)

$\mathrm{S} \in \mathit{Sort} = \mathit{base} \mid \mathrm{S}_1 \Rightarrow \mathrm{S}_2$

$\lambda : \forall\{\mathrm{S}, \mathrm{T}\}.\mathit{Op}\,(\mathrm{S} \Rightarrow \mathrm{T})$

$\$: \forall\{\mathrm{S}, \mathrm{T}\}.\mathit{Op}\,\mathrm{T}$

$\mathit{arity}\,\lambda\{\mathrm{S}, \mathrm{T}\} = [([\mathrm{S}], \mathrm{T})]$

$\mathit{arity}\,\$\{\mathrm{S}, \mathrm{T}\} = [([], \mathrm{S} \Rightarrow \mathrm{T}), ([], \mathrm{S})]$

Free monad F^*:

$$\frac{\mathit{tbody} \in F^*\,X(\Gamma \dotplus [\mathrm{S}] \vdash \mathrm{T})}{\lambda\,\mathit{tbody} \in F^*\,X(\Gamma \vdash \mathrm{S} \Rightarrow \mathrm{T})}$$

$$\frac{\mathit{tf} \in F^*\,X\,(\Gamma \dotplus [] \vdash \mathrm{S} \Rightarrow \mathrm{T}) \quad \mathit{tx} \in F^*\,X\,(\Gamma \dotplus [] \vdash \mathrm{S})}{\mathit{tf}\,\$\,\mathit{tx} \in F^*\,X\,(\Gamma \vdash \mathrm{T})} \qquad \frac{x \in X\,\mathit{jud}}{x \in F^*\,X\,\mathit{jud}}$$

Second Order Multisorted Algebraic Theories (SOMATs)

Eq-free SOMAT

Fix $\mathit{Sort} : \mathbf{Set}$

$\mathit{Ctx} := \mathit{Tele} := \mathbf{List}\,\mathit{Sort}$,
$\mathit{Jud} := \mathit{Ctx} \times \mathit{Sort}$.

$\mathit{Op} : \mathit{Sort} \to \mathbf{Set}$
$\mathit{arity} : \forall\{S\}.\mathit{Op}\,S \to \mathbf{List}\,(\mathit{Tele} \times \mathit{Sort})$

Syntax functor $F : \mathbf{Set}^{\mathit{Jud}} \to \mathbf{Set}^{\mathit{Jud}}$:
$F\,X\,(\Gamma \vdash S) = \Sigma(\mathit{op} : \mathit{Op}\,S).$
$\mathbf{ListP}_{\mathit{arity}\;\mathit{op}}\,(\lambda(\Theta, T).X\,(\Gamma \dotplus \Theta \vdash T)).$

Term monad $F^*\,X\,\mathit{jud} \cong X\,\mathit{jud} \uplus F\,(F^*\,X)\,\mathit{jud}$.

Model = F-algebra with variables
= F^*-monad-algebra with variables.

Example (STLC)

$S \in \mathit{Sort} = \mathit{base} \mid S_1 \Rightarrow S_2$

$\lambda : \forall\{S, T\}.\mathit{Op}\,(S \Rightarrow T)$
$\$: \forall\{S, T\}.\mathit{Op}\,T$

$\mathit{arity}\,\lambda\{S, T\} = [([S], T)]$
$\mathit{arity}\,\$\{S, T\} = [([], S \Rightarrow T), ([], S)]$

Free monad F^*:

$$\frac{\mathit{tbody} \in F^*\,X(\Gamma \dotplus [S] \vdash T)}{\lambda\,\mathit{tbody} \in F^*\,X(\Gamma \vdash S \Rightarrow T)}$$

$$\frac{\mathit{tf} \in F^*\,X\,(\Gamma \dotplus [] \vdash S \Rightarrow T) \quad \mathit{tx} \in F^*\,X\,(\Gamma \dotplus [] \vdash S)}{\mathit{tf}\,\$\,\mathit{tx} \in F^*\,X\,(\Gamma \vdash T)} \qquad \frac{x \in X\,\mathit{jud}}{x \in F^*\,X\,\mathit{jud}}$$

SOMATs are not good enough!

SOMATs:

- Simply typed, but see
  - Fiore (2008)
  - Bocquet, Kaposi & Sattler (2021)
- Non-modal:
  - Contexts are **lists** of sorts,
  - Substitutions are **lists** of terms.
- Admissible substitution because $\Gamma \mapsto \Gamma \dotplus \Theta$ is **functorial!** $(\lambda\, tbody)[\sigma] = \lambda(tbody[\sigma \dotplus [S]])$

Menkar / MTT (Multimodal Type Theory):

- Dependently typed
- Modalities μ where $[\![🔒_\mu]\!] \dashv [\![\mu]\!]$

$$\frac{\Gamma\ \mathbf{ctx}}{\Gamma, 🔒_\mu\ \mathbf{ctx}} \qquad \frac{\Gamma, 🔒_\mu \vdash t : T}{\Gamma \vdash \mathbf{mod}_\mu\, t : \langle \mu \mid T \rangle}$$

$$\frac{\sigma : \Gamma \to \Delta \qquad \alpha : \nu \Rightarrow \mu}{(\sigma, 🔒_\alpha) : (\Gamma, 🔒_\mu) \to (\Delta, 🔒_\nu)}$$

- Admissible substitution because $\Gamma \mapsto (\Gamma, 🔒_\mu)$ is **functorial!** $(\mathbf{mod}_\mu\, t)[\sigma] = \mathbf{mod}_\mu\, (t[\sigma, 🔒_\mu])$

SOMATs are not good enough!

SOMATs:

- Simply typed, but see
  - Fiore (2008)
  - Bocquet, Kaposi & Sattler (2021)
- Non-modal:
  - Contexts are **lists** of sorts,
  - Substitutions are **lists** of terms.
- Admissible substitution because $\Gamma \mapsto \Gamma \dotplus \Theta$ is **functorial!** $(\lambda\, tbody)[\sigma] = \lambda(tbody[\sigma \dotplus [S]])$

Menkar / MTT (Multimodal Type Theory):

- Dependently typed
- Modalities μ where $[\![🔒_\mu]\!] \dashv [\![\mu]\!]$

$$\frac{\Gamma\ \mathbf{ctx}}{\Gamma, 🔒_\mu\ \mathbf{ctx}} \qquad \frac{\Gamma, 🔒_\mu \vdash t : T}{\Gamma \vdash \mathbf{mod}_\mu\, t : \langle \mu \mid T \rangle}$$

$$\frac{\sigma : \Gamma \to \Delta \qquad \alpha : \nu \Rightarrow \mu}{(\sigma, 🔒_\alpha) : (\Gamma, 🔒_\mu) \to (\Delta, 🔒_\nu)}$$

- Admissible substitution because $\Gamma \mapsto (\Gamma, 🔒_\mu)$ is **functorial!** $(\mathbf{mod}_\mu\, t)[\sigma] = \mathbf{mod}_\mu\,(t[\sigma, 🔒_\mu])$

SOMATs are not good enough!

SOMATs:

- Simply typed, but see
  - Fiore (2008)
  - Bocquet, Kaposi & Sattler (2021)
- Non-modal:
  - Contexts are **lists** of sorts,
  - Substitutions are **lists** of terms.
- Admissible substitution because $\Gamma \mapsto \Gamma \dotplus \Theta$ is **functorial!** $(\lambda\, tbody)[\sigma] = \lambda(tbody[\sigma \dotplus [S]])$

Menkar / MTT (Multimodal Type Theory):

- Dependently typed
- Modalities μ where $[\![🔒_\mu]\!] \dashv [\![\mu]\!]$

$$\frac{\Gamma\ \mathbf{ctx}}{\Gamma, 🔒_\mu\ \mathbf{ctx}} \qquad \frac{\Gamma, 🔒_\mu \vdash t : T}{\Gamma \vdash \mathbf{mod}_\mu\, t : \langle \mu \mid T \rangle}$$

$$\frac{\sigma : \Gamma \to \Delta \qquad \alpha : \nu \Rightarrow \mu}{(\sigma, 🔒_\alpha) : (\Gamma, 🔒_\mu) \to (\Delta, 🔒_\nu)}$$

- Admissible substitution because $\Gamma \mapsto (\Gamma, 🔒_\mu)$ is **functorial!** $(\mathbf{mod}_\mu\, t)[\sigma] = \mathbf{mod}_\mu\, (t[\sigma, 🔒_\mu])$

SOMATs are not good enough!

SOMATs:

- Simply typed, but see
  - Fiore (2008)
  - Bocquet, Kaposi & Sattler (2021)
- Non-modal:
  - Contexts are **lists** of sorts,
  - Substitutions are **lists** of terms.
- Admissible substitution because $\Gamma \mapsto \Gamma \dotplus \Theta$ is **functorial!** $(\lambda\, tbody)[\sigma] = \lambda(tbody[\sigma \dotplus [S]])$

Menkar / MTT (Multimodal Type Theory):

- Dependently typed
- Modalities μ where $[\![🔒_\mu]\!] \dashv [\![\mu]\!]$

$$\frac{\Gamma\ \mathbf{ctx}}{\Gamma, 🔒_\mu\ \mathbf{ctx}} \qquad \frac{\Gamma, 🔒_\mu \vdash t : T}{\Gamma \vdash \mathbf{mod}_\mu\, t : \langle \mu \mid T \rangle}$$

$$\frac{\sigma : \Gamma \to \Delta \qquad \alpha : \nu \Rightarrow \mu}{(\sigma, 🔒_\alpha) : (\Gamma, 🔒_\mu) \to (\Delta, 🔒_\nu)}$$

- Admissible substitution because $\Gamma \mapsto (\Gamma, 🔒_\mu)$ is **functorial!** $(\mathbf{mod}_\mu\, t)[\sigma] = \mathbf{mod}_\mu\, (t[\sigma, 🔒_\mu])$

SOMATs are not good enough!

SOMATs:

- Simply typed, but see
  - Fiore (2008)
  - Bocquet, Kaposi & Sattler (2021)
- Non-modal:
  - Contexts are **lists** of sorts,
  - Substitutions are **lists** of terms.
- Admissible substitution because $\Gamma \mapsto \Gamma \dotplus \Theta$ is **functorial!** $(\lambda\, tbody)[\sigma] = \lambda(tbody[\sigma \dotplus [S]])$

Menkar / MTT (Multimodal Type Theory):

- Dependently typed
- Modalities μ where $[\![🔒_\mu]\!] \dashv [\![\mu]\!]$

$$\frac{\Gamma\ \mathbf{ctx}}{\Gamma, 🔒_\mu\ \mathbf{ctx}} \qquad \frac{\Gamma, 🔒_\mu \vdash t : T}{\Gamma \vdash \mathbf{mod}_\mu\, t : \langle \mu \mid T \rangle}$$

$$\frac{\sigma : \Gamma \to \Delta \qquad \alpha : \nu \Rightarrow \mu}{(\sigma, 🔒_\alpha) : (\Gamma, 🔒_\mu) \to (\Delta, 🔒_\nu)}$$

- Admissible substitution because $\Gamma \mapsto (\Gamma, 🔒_\mu)$ is **functorial!** $(\mathbf{mod}_\mu\, t)[\sigma] = \mathbf{mod}_\mu\, (t[\sigma, 🔒_\mu])$

SOMATs are not good enough!

SOMATs:

- Simply typed, but see
  - Fiore (2008)
  - Bocquet, Kaposi & Sattler (2021)
- Non-modal:
  - Contexts are **lists** of sorts,
  - Substitutions are **lists** of terms.
- Admissible substitution because
  $\Gamma \mapsto \Gamma \dotplus \Theta$ is **functorial!**
  $(\lambda\, tbody)[\sigma] = \lambda(tbody[\sigma \dotplus [S]])$

Menkar / MTT (Multimodal Type Theory):

- Dependently typed
- Modalities μ where $[\![\text{🔒}_\mu]\!] \dashv [\![\mu]\!]$

$$\frac{\Gamma\ \mathbf{ctx}}{\Gamma, \text{🔒}_\mu\ \mathbf{ctx}} \qquad \frac{\Gamma, \text{🔒}_\mu \vdash t : T}{\Gamma \vdash \mathbf{mod}_\mu\, t : \langle \mu \mid T \rangle}$$

$$\frac{\sigma : \Gamma \to \Delta \qquad \alpha : \nu \Rightarrow \mu}{(\sigma, \text{🔒}_\alpha) : (\Gamma, \text{🔒}_\mu) \to (\Delta, \text{🔒}_\nu)}$$

- Admissible substitution because
  $\Gamma \mapsto (\Gamma, \text{🔒}_\mu)$ is **functorial!**
  $(\mathbf{mod}_\mu\, t)[\sigma] = \mathbf{mod}_\mu\, (t[\sigma, \text{🔒}_\mu])$

Other systems

Dual contexts

Pfenning & Davies (2001)

$$\frac{\Delta\,;\cdot \vdash t : T}{\Delta\,;\Gamma \vdash \mathbf{box}\,t : \Box\,T}$$

$(\Delta\,;\Gamma) \mapsto (\Delta\,;\cdot)$ is **functorial!**

$(\mathbf{box}\,t)[\delta\,;\gamma] = \mathbf{box}\,(t[\delta;\cdot])$

$(\mathbb{I} \to -) \dashv \sqrt[\mathbb{I}]{-}$ (Amazing right adjoint)

Licata, Orton, Pitts & Spitters (2018)

$$\frac{\mathbb{I} \to \Gamma \vdash t : T}{\Gamma \vdash \mathbf{amaze}\,t : \sqrt[\mathbb{I}]{T}}$$

$\Gamma \mapsto (\mathbb{I} \to \Gamma)$ is **functorial!**

$(\mathbf{amaze}\,t)[\sigma] = \mathbf{amaze}\,(t[(\sigma \circ -)])$

Other systems

Dual contexts

Pfenning & Davies (2001)

$$\frac{\Delta\,;\cdot \vdash t : T}{\Delta\,;\Gamma \vdash \mathbf{box}\, t : \Box\, T}$$

$(\Delta\,;\Gamma) \mapsto (\Delta\,;\cdot)$ is **functorial!**

$(\mathbf{box}\, t)[\delta\,;\gamma] = \mathbf{box}\,(t[\delta;\cdot])$

$(\mathbb{I} \to -) \dashv \sqrt[\mathbb{I}]{-}$ (Amazing right adjoint)

Licata, Orton, Pitts & Spitters (2018)

$$\frac{\mathbb{I} \to \Gamma \vdash t : T}{\Gamma \vdash \mathbf{amaze}\, t : \sqrt[\mathbb{I}]{T}}$$

$\Gamma \mapsto (\mathbb{I} \to \Gamma)$ is **functorial!**

$(\mathbf{amaze}\, t)[\sigma] = \mathbf{amaze}\,(t[(\sigma \circ -)])$

Trivially Admissible Substitution

Typing rules have the form

$$\frac{\Gamma.\Phi_i \vdash t_i : T_i}{\Gamma \vdash \mathbf{op}(t_i)_i : T}$$

where $-.\Phi_i$ acts **functorially!**

$\mathbf{op}(t_i)_i[\sigma] = \mathbf{op}(t_i[\sigma.\Phi_i])_i$

Call Φ_i a **Junctor**:

- Generalizes **binder** (Lat.: iunctor),
- Sounds a lot like **functor!**

Gist of the Talk

Trivially Admissible Substitution

Typing rules have the form

$$\frac{\Gamma.\Phi_i \vdash t_i : T_i}{\Gamma \vdash \mathbf{op}(t_i)_i : T}$$

where $-.\Phi_i$ acts **functorially!**
$\mathbf{op}(t_i)_i[\sigma] = \mathbf{op}(t_i[\sigma.\Phi_i])_i$

Call Φ_i a **Junctor**:

- Generalizes **binder** (Lat.: iunctor),
- Sounds a lot like **functor!**

Contextual Multisorted Algebraic Theories (CMATs)

Eq-free SOMAT

Fix $\mathit{Sort} : \mathbf{Set}$

$\mathit{Ctx} := \mathit{Tele} := \mathbf{List}\,\mathit{Sort}$,

$\mathit{Jud} := \mathit{Ctx} \times \mathit{Sort}$.

(Fin.) Container:

$\mathit{Op} : \mathit{Sort} \to \mathbf{Set}$

$\mathit{arity} : \forall\{\mathrm{S}\}.\mathit{Op}\,\mathrm{S} \to \mathbf{List}\,(\mathit{Tele} \times \mathit{Sort})$

Eq-free CMAT

Fix $\mathit{Sort} : \mathbf{Set}$

Fix $\mathit{Ctx}, \mathit{Junctor} : \mathbf{Set}$,

$\mathit{Jud} := \mathit{Ctx} \times \mathit{Sort}$.

(Fin.) Container:

$\mathit{Op} : \mathit{Sort} \to \mathbf{Set}$

$\mathit{arity} : \forall\{\mathrm{S}\}.\mathit{Op}\,\mathrm{S} \to \mathbf{List}\,(\mathit{Junctor} \times \mathit{Sort})$

Example (Amazing right adjoint)

$\mathbf{amaze} : \forall \mathrm{S}.\mathit{Op}(\sqrt[\mathbb{I}]{\mathrm{S}})$

$\mathit{arity}\,\mathbf{amaze} = [(\mathbb{I} \Rightarrow -, \mathrm{S})]$

$$\frac{t \in F^*\,X(\mathbb{I} \Rightarrow \Gamma \vdash \mathrm{S})}{\mathbf{amaze}\,t \in F^*\,X(\Gamma \vdash \sqrt[\mathbb{I}]{\mathrm{S}})}$$

Contextual Multisorted Algebraic Theories (CMATs)

Eq-free SOMAT

Fix *Sort* : **Set**

$\mathit{Ctx} := \mathit{Tele} := \textbf{List}\,\mathit{Sort}$,

$\mathit{Jud} := \mathit{Ctx} \times \mathit{Sort}$.

(Fin.) Container:

$\mathit{Op} : \mathit{Sort} \to \textbf{Set}$

$\mathit{arity} : \forall\{S\}.\mathit{Op}\,S \to \textbf{List}\,(\mathit{Tele} \times \mathit{Sort})$

Eq-free CMAT

Fix *Sort* : **Set**

Fix $\mathit{Ctx}, \mathit{Junctor} : \textbf{Set}$,

$\mathit{Jud} := \mathit{Ctx} \times \mathit{Sort}$.

(Fin.) Container:

$\mathit{Op} : \mathit{Sort} \to \textbf{Set}$

$\mathit{arity} : \forall\{S\}.\mathit{Op}\,S \to \textbf{List}\,(\mathit{Junctor} \times \mathit{Sort})$

Example (Amazing right adjoint)

$\textbf{amaze} : \forall S.\mathit{Op}(\sqrt[\mathbb{I}]{S})$

$\mathit{arity}\,\textbf{amaze} = [(\mathbb{I} \Rightarrow -, S)]$

$$\frac{t \in F^*\,X(\mathbb{I} \Rightarrow \Gamma \vdash S)}{\textbf{amaze}\,t \in F^*\,X(\Gamma \vdash \sqrt[\mathbb{I}]{S})}$$

Contextual Multisorted Algebraic Theories (CMATs)

Eq-free SOMAT

Fix $Sort$: **Set**

$Ctx := Tele := \mathbf{List}\, Sort$,

$Jud := Ctx \times Sort$.

(Fin.) Container:

$Op : Sort \to \mathbf{Set}$

$arity : \forall\{S\}.Op\, S \to \mathbf{List}\,(Tele \times Sort)$

Eq-free CMAT

Fix $Sort$: **Set**

Fix $Ctx, Junctor$: **Set**,

$Jud := Ctx \times Sort$.

(Fin.) Container:

$Op : Sort \to \mathbf{Set}$

$arity : \forall\{S\}.Op\, S \to \mathbf{List}\,(Junctor \times Sort)$

Example (Amazing right adjoint)

$\mathbf{amaze} : \forall S.Op(\sqrt[\mathbb{I}]{S})$

$arity\,\mathbf{amaze} = [(\mathbb{I} \Rightarrow -, S)]$

$$\frac{t \in F^* X(\mathbb{I} \Rightarrow \Gamma \vdash S)}{\mathbf{amaze}\, t \in F^* X(\Gamma \vdash \sqrt[\mathbb{I}]{S})}$$

Status of <u>cubical Agda formalization</u>:

- ✔ MATs & their models
- ⌛ CMATs
  - ✔ Eq-free CMATs & MAT translations
  - ☐ Eq theories & MAT translations
  - ⌛ Adapt FS22's substitution theorem
  - ☐ Characterize models
- ☐ Generalize AACMM21's results
- ☐ SOMATs are CMATs
- ☐ M(S)TT is a CMAT
- ☐ NbE via categorical gluing
- ? Wrap this up as a programming assistant?

- ⌛ Contributing to Agda cubical library

Status of cubical Agda formalization:

- ✔ MATs & their models
- ⌛ CMATs
  - ✔ Eq-free CMATs & MAT translations
  - ☐ Eq theories & MAT translations
  - ⌛ Adapt FS22's substitution theorem
  - ☐ Characterize models
- ☐ Generalize AACMM21's results
- ☐ SOMATs are CMATs
- ☐ M(S)TT is a CMAT
- ☐ NbE via categorical gluing
- ? Wrap this up as a programming assistant?

⌛ Contributing to Agda cubical library

Status and Plans

Status of cubical Agda formalization:

- ✔ MATs & their models
- ⌛ CMATs
  - ✔ Eq-free CMATs & MAT translations
  - ☐ Eq theories & MAT translations
  - ⌛ Adapt FS22's substitution theorem
  - ☐ Characterize models
- ☐ Generalize AACMM21's results
- ☐ SOMATs are CMATs
- ☐ M(S)TT is a CMAT
- ☐ NbE via categorical gluing
- ? Wrap this up as a programming assistant?

- ⌛ Contributing to Agda cubical library

Status and Plans

Status of cubical Agda formalization:

- ✔ MATs & their models
- ⌛ CMATs
  - ✔ Eq-free CMATs & MAT translations
  - ☐ Eq theories & MAT translations
  - ⌛ Adapt FS22's substitution theorem
  - ☐ Characterize models
- ☐ Generalize AACMM21's results
- ☐ SOMATs are CMATs
- ☐ M(S)TT is a CMAT
- ☐ NbE via categorical gluing
- ? Wrap this up as a programming assistant?

⌛ Contributing to Agda cubical library

Status and Plans

Status of cubical Agda formalization:

- ✔ MATs & their models
- ⌛ CMATs
  - ✔ Eq-free CMATs & MAT translations
  - ☐ Eq theories & MAT translations
  - ⌛ Adapt FS22's substitution theorem
  - ☐ Characterize models
- ☐ Generalize AACMM21's results
- ☐ SOMATs are CMATs
- ☐ M(S)TT is a CMAT
- ☐ NbE via categorical gluing
- ? Wrap this up as a programming assistant?

⌛ Contributing to Agda cubical library

Status of <u>cubical Agda formalization</u>:

- ✔ MATs & their models
- ⌛ CMATs
  - ✔ Eq-free CMATs & MAT translations
  - ☐ Eq theories & MAT translations
  - ⌛ Adapt FS22's substitution theorem
  - ☐ Characterize models
- ☐ Generalize AACMM21's results
- ☐ SOMATs are CMATs
- ☐ M(S)TT is a CMAT
- ☐ NbE via categorical gluing
- ? Wrap this up as a programming assistant?

- ⌛ Contributing to Agda cubical library

Status and Plans

Status of cubical Agda formalization:

- ✔ MATs & their models
- ⧗ CMATs
  - ✔ Eq-free CMATs & MAT translations
  - ☐ Eq theories & MAT translations
  - ⧗ Adapt FS22's substitution theorem
  - ☐ Characterize models
- ☐ Generalize AACMM21's results
- ☐ SOMATs are CMATs
- ☐ M(S)TT is a CMAT
- ☐ NbE via categorical gluing
- ? Wrap this up as a programming assistant?

- ⧗ Contributing to Agda cubical library

Thanks!

Questions?