

Modular abstract syntax trees (MAST): substitution tensors with second-class sorts

Marcelo Fiore, Kajetan Granops, Mihail-Codrin Iftode, Ohad Kammar,
Georg Moser, and Sam Staton

Paper:



Slides:



Theoretical Computer Science Seminar

24 October 2025

Department of Computer Science, University of Birmingham, England

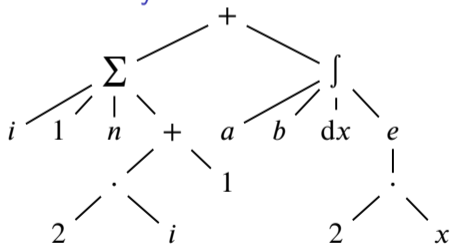
Syntax representation

$$\left(\sum_{i=1}^n (2i + 1) \right) + \int_a^b e^{ax} dx$$

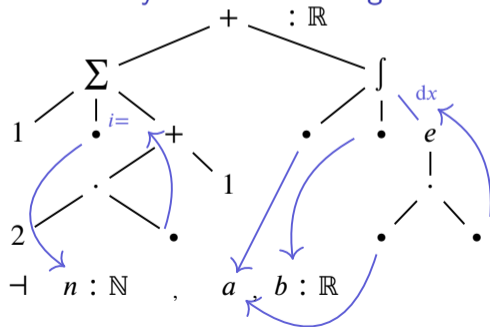
Concrete syntax

"(", " Σ ", "-", "{", "i", "=", "1", "}", "{", "n", "(", "2", "i", "+", "1", ")", "}", ...

Abstract syntax



Abstract syntax with binding



Call-by-Value λ -calculus

$A, B, C ::=$	type	$V, W ::=$	value
β	base	x	variable
$ A \rightarrow B$	function	$ \lambda x : A. M$	function abst.
$ \langle C_i : A_i \mid i \in I \rangle$	record (I finite)	$ (C_i : V_i \mid i \in I)$	record c'tor
$ \{ C_i : A_i \mid i \in I \}$	variant (I finite)	$ A.C_i V$	variant c'tor
\vdots		\vdots	
$M, N, K, L ::=$	term		
$\text{val } V$	value		
$ \text{let } x_1 = M_1; \dots; x_n = M_n \text{ in } N$	sequencing		
$ M @ N$	function application		
$ (C_1 : M_1, \dots, C_n : M_n)$	record constructor		
$ \text{case } M \text{ of } (C_1 x_1, \dots, C_n x_n) \Rightarrow N$	record pattern match		
$ A.C_i M$	variant constructor		
$ \text{case } M \text{ of } \{ C_i x_i \Rightarrow M_i \mid i \in I \} N$	variant pattern match		
\vdots			

High-level motivation

Initial Algebra Semantics Programme

[Goguen and Thatcher'74]

Denotational semantics á la carte

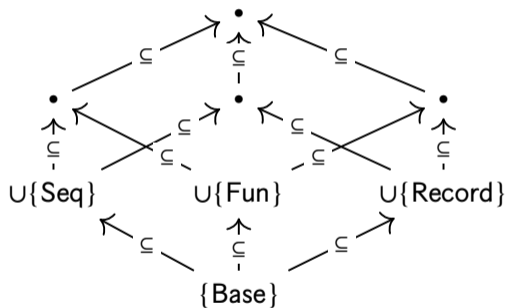
[homage to Swierstra'08, Forster and Stark'20]

CBV customisation menu

fragment	syntactic constructs	types	semantics
base	returning a value: val		strong monad over a Cartesian category
sequential	sequencing: let		
functions	abst., app. $(\lambda x. : A), (@)$	function (\rightarrow)	Kleisli exponentials
variants	c'tors, pattern match $A.C_i-$, case – of $\{C_i x_i \Rightarrow - \mid i \in I\}$	variant $\{C_i : - \mid i \in I\}$	distributive category
⋮			

Iterative semantic development

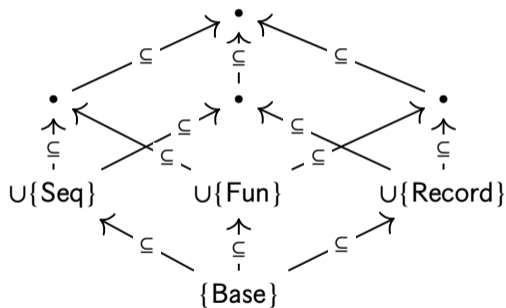
- Add syntax
- Add semantics



- Profit!

Iterative semantic development

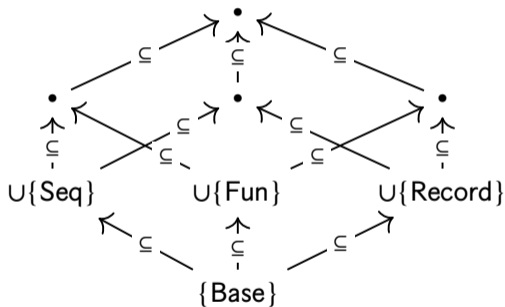
- ▶ Add syntax
- ▶ Add semantics
- ▶ Develop meta-theory:
 - ▶ Substitution lemma
 - ▶ Compositionality
 - ▶ Soundness
 - ▶ Adequacy
- ▶ Profit!



Dream vs. **Bleak** Reality

Iterative semantic development

- ▶ Add syntax
- ▶ Add semantics
- ▶ Develop meta-theory:
 - ▶ Substitution lemma
Tedious and boring
 - ▶ Compositionality
Tedious and boring
 - ▶ Soundness
 - ▶ Adequacy
- ▶ Profit!



Meta-theory: the tedious parts

Lemma (substitution)

Syntactic substitution corresponds to semantic composition:

$$\llbracket M [\theta] \rrbracket = \llbracket M \rrbracket \circ \llbracket \theta \rrbracket$$

Lemma (compositionality)

Composite semantics is independent of component syntax:

$$\llbracket C[M] \rrbracket = \text{plug}(\llbracket C[-] \rrbracket, \llbracket M \rrbracket)$$

Meta-theory: the tedious parts

Lemma (substitution)

Syntactic substitution corresponds to semantic composition:

$$\llbracket M [\theta] \rrbracket = \llbracket M \rrbracket \circ \llbracket \theta \rrbracket$$

Proof.

Presupposes a syntactic substitution lemma. Typically several inductions over all constructs. □

Lemma (compositionality)

Composite semantics is independent of component syntax:

$$\llbracket C[M] \rrbracket = \text{plug}(\llbracket C[-] \rrbracket, \llbracket M \rrbracket)$$

Proof.

Tediously define terms with holes, plugging holes syntactically, carefully capturing some variables but not others. Then induction over semantics. □

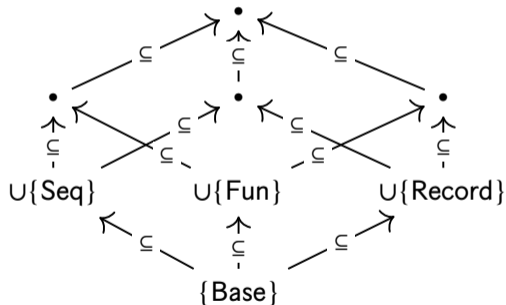
It would be nice if tedious bits were...
... free

Dream vs. Reality

It would be nice if tedious bits were...

... free

... syntactically scaleable: additive syntactic work per new feature



Spec

[Wadler'98]

Both:

- ▶ **Extend** object-language syntax
- ▶ **Add** meta-language functions/properties of programs

While:

- ▶ Without recompiling previous modules; alternatively
- ▶ Retaining and reusing both old and new languages

Some solutions

- ▶ Scala Mixings [Zenger'98, Zenger and Odersky'01]
- ▶ Visitor Pattern in Pizza, Zodiac [Krishnamurthi, Felleisen and Friedman'98]
- ▶ Recursive Generics [Wadler'98]
- ▶ **Data-types á la carte**: coproducts of signature functors [Swierstra'08]

- ▶ Initial algebra characterisation for abstract syntax with binding-aware substitution
- ▶ Robust to extensions:
 - ▶ polymorphism
 - ▶ mechanisation
 - ▶ substructurality
- ▶ CBN works smoothly. Doesn't cover CBV. Technical reasons later:
 - ▶ Substitute **in**: values and terms
 - ▶ Substitute for variables: values only

[Fiore and Hamana'13]
[Crole'11, Allais et al.'18,
Fiore and Szamozvancev'22]
[Fiore and Ranchod'25]

Slogan

for substitution: values are 1st-class but terms are 2nd-class
[cf. Levy's CBPV, '04]

Goal: abstract syntax with heterogeneous sorting

Sorting signature \mathbf{R}

- ▶ set sort

partitioned into

- ▶ bindable/ 1^{st} -class sorts
 $s \in \text{Bind}$
- ▶ non-bindable/ 2^{nd} -class sorts

Example (CBV sorting signature)

- ▶ $\text{sort} := \{A, \text{comp } A \mid A \in \text{Type}\}$
- ▶ $\text{Bind} := \text{Type}_{\text{CBV}}$

Example (CBPV sorting signature)

- ▶ $\text{Bind} := \{A \mid A \text{ value type}\}$
- ▶ $\text{sort} := \text{Type}_{\text{CBPV}}$

classical theory (SOAS)

PSh (sort \times sort_⊥), \otimes
monoidal product

generalise
 \rightsquigarrow

this work (MAST)

PSh (sort \times Bind_⊥), \otimes
right-unital associative
skew monoidal product

Skew monoidal heterogeneous tensor

$$(P \otimes Q) \otimes L \cong P \otimes (Q \otimes L) \quad (\text{associative})$$

$$P \otimes \mathbb{I} \cong P \quad (\text{right-unital})$$

$$\mathbb{I} \otimes Q \xrightarrow{\ell} Q \quad (\text{non-invertible!})$$

$$(\mathbb{I} \otimes 1)_s = \emptyset \not\cong 1_s (s \notin \text{Bind})$$

Modular Abstract Syntax Trees (MAST)

- ▶ SOAS $\xrightarrow{\text{generalise}}$ 2nd-class sorts
Using **skew** bicategories/monoidal categories, and:
 - ▶ Kleisli bicategories [Gambino, Fiore, Hyland, and Winskel'19]
 - ▶ Familial theory of SOAS [Fiore and Szamozvancev'25]
- ▶ MAST tutorial
- ▶ Case-study: CBV semantics á la carte (128 substitution lemmata)

WIP

- ▶ Idris 2 implementation of computational fragment [cf. Fiore and Szamozvancev'22]
Case-study: intrinsically-typed FFI-binding with holes for SMTLIB (29 theories)
- ▶ Replace skew monoidal structure and monoids with
monoidal structure and actions

[cf. Fiore and Turi'01]

Talk structure

- ▶ Contribution
- ▶ Substitution monoids
- ▶ MAST in detail
- ▶ WIP

Capstone: abstract syntax and substitution universality

Thm (representation)

*abstract syntax with operators in \mathbf{O} and holes in \mathbf{H}
amounts to
free substitution \mathbf{O} -monoid over \mathbf{H} :*

$$\begin{array}{ccc} & \mathbf{H} & \\ & \downarrow \text{?}[-] & \\ \mathbf{SH} \otimes \mathbf{SH} & \xrightarrow{-[-]} \mathbf{SH} & \xleftarrow[\text{var}]{\mathbb{I}} \mathbf{SH} \\ & \uparrow [-] & \\ & \mathbf{O}(\mathbf{SH}) & \end{array}$$

Plugging holes/metavariable substitution

Kleisli extension ($\gg=$) for \mathbf{O} -monoid monad.

Key propaganda

compositional, binding-respecting denotational semantics
amounts to
substitution **O**-monoid:

$$\begin{array}{ccc} \mathbf{M} \otimes \mathbf{M} & \xrightarrow{-[-]} & \mathbf{M} \\ & \llbracket - \rrbracket \uparrow & \longleftarrow \mathbb{I} \\ & \mathbf{OM} & \text{var} \end{array}$$

The denotational semantics for terms with holes in **H** is the unique substitution **O**-monoid homomorphism over **H**:

$$(\$ \mathbf{H}, -[-], \text{var}, \llbracket - \rrbracket, ?-[\text{id}]) \xrightarrow{\llbracket - \rrbracket} (\mathbf{M}, -[-], \text{var}, \llbracket - \rrbracket, \text{menv}) \quad (\mathbf{H} \xrightarrow{\text{menv}} \mathbf{M})$$

Lemma (substitution)

Syntactic substitution corresponds to semantic composition:

$$\llbracket M [\theta] \rrbracket = \llbracket M \rrbracket \circ \llbracket \theta \rrbracket$$

Lemma (compositionality)

Composite semantics is independent of component syntax:

$$\begin{aligned} \llbracket C[M] \rrbracket = \\ \text{plug}(\llbracket C[-] \rrbracket, \llbracket M \rrbracket) \end{aligned}$$

Meta-theory in one line

Lemma (substitution)

Syntactic substitution corresponds to semantic composition:

substitution monoid homomorphism

$$\begin{array}{c} \downarrow \\ \llbracket M [\theta] \rrbracket = \llbracket -[-] [M, \theta] \rrbracket = -[-] [\llbracket M \rrbracket, \llbracket \theta \rrbracket] := \llbracket M \rrbracket \circ \llbracket \theta \rrbracket \end{array}$$

Lemma (compositionality)

Composite semantics is independent of component syntax:

$$\begin{aligned} \llbracket C[M] \rrbracket = \\ \text{plug}(\llbracket C[-] \rrbracket, \llbracket M \rrbracket) \end{aligned}$$

Meta-theory in one line

Lemma (substitution)

Syntactic substitution corresponds to semantic composition:

$$\begin{array}{c} \text{substitution monoid homomorphism} \\ \downarrow \\ \llbracket M [\theta] \rrbracket = \llbracket -[-] [M, \theta] \rrbracket = -[-] [\llbracket M \rrbracket, \llbracket \theta \rrbracket] := \llbracket M \rrbracket \circ \llbracket \theta \rrbracket \end{array}$$

Lemma (compositionality)

Composite semantics is independent of component syntax:

$$\begin{aligned} \llbracket C[M] \rrbracket &= \llbracket C[?m] \gg (m \mapsto M) \rrbracket = \llbracket C[?m] \rrbracket \gg (m \mapsto \llbracket M \rrbracket) \\ &=: \text{plug}(\llbracket C[-] \rrbracket, \llbracket M \rrbracket) \overset{\uparrow}{\gg}_{\text{is}} \end{aligned}$$

homomorphic extension

Talk structure

- ▶ Contribution
- ▶ Substitution monoids
- ▶ **MAST in detail**
- ▶ WIP

MAST summary: semantic domain for syntax and semantics

MAST provides ($\mathbf{R} = (\text{sort}, \text{Bind})$ sorting system)

► Contexts

$$\text{Bind}_{\vdash} \ni \Gamma ::= [x_1 : s_1, \dots, x_n : s_n]$$

► Renamings

$$\text{Bind}_{\vdash}(\Gamma, \Delta) \ni \Gamma \vdash \rho : \Delta$$

► \mathbf{R} -structures:

$$\mathbf{PSh}(\text{sort} \times \text{Bind}_{\vdash}) \ni \mathbf{P} : \text{sort} \times \text{Bind}_{\vdash}^{\text{op}} \rightarrow \mathbf{Set}$$

$$P_s \Gamma \ni p: \quad \text{sort } s \text{ element} \quad \text{with variables in } \Gamma$$

► Variables structure:

$$\mathbf{R}\text{-Struct} \ni \mathbb{I}_s \Gamma := \{x | (x : s) \in \Gamma\}$$

► substitution tensors:

$$\mathbf{R}\text{-Struct} \ni \mathbf{P} \otimes \mathbf{Q}, \mathbf{P} \otimes \cdot \left(\text{var} \begin{array}{c} \mathbb{I} \\ \downarrow \\ \mathbf{A} \end{array} \right)$$

$$(\mathbf{P} \otimes \mathbf{Q})_s \Gamma \ni [p, \theta]_{\Delta}:$$

$$P\text{-element: } p \in P_s \Delta$$

$$Q\text{-closure: } \theta \in \prod_{(y:r) \in \Delta} Q_r \Gamma$$

identifying, e.g.:

$$\begin{aligned} [p[\text{weaken}], \theta]_{\Delta_1 \# \Delta_2} &= [p, \theta \circ \rho]_{\Delta_1} \\ [p[x', x'' \mapsto x]_{x \in \Delta}, \theta]_{\Delta} &= [p, \theta \# \theta]_{\Delta \# \Delta} \end{aligned}$$

MAST summary: semantic domain for syntax and semantics

MAST provides ($\mathbf{R} = (\text{sort}, \text{Bind})$ sorting system)

► Contexts

$$\text{Bind}_{\vdash} \ni \Gamma ::= [x_1 : s_1, \dots, x_n : s_n]$$

► Renamings

$$\text{Bind}_{\vdash}(\Gamma, \Delta) \ni \Gamma \vdash \rho : \Delta$$

► \mathbf{R} -structures:

$$\mathbf{PSh}(\text{sort} \times \text{Bind}_{\vdash}) \ni P : \text{sort} \times \text{Bind}_{\vdash}^{\text{op}} \rightarrow \mathbf{Set}$$

$$P_s \Gamma \ni p: \quad \text{sort } s \text{ element} \quad \text{with variables in } \Gamma$$

► Variables structure:

$$\mathbf{R}\text{-Struct} \ni \mathbb{I}_s \Gamma := \{x | (x : s) \in \Gamma\}$$

► substitution tensors:

$$\mathbf{R}\text{-Struct} \ni P \otimes Q, P \otimes \cdot \left(\text{var} \begin{array}{c} \mathbb{I} \\ \downarrow \\ A \end{array} \right)$$

$$(P \otimes Q)_s \Gamma \ni [p, \theta]_{\Delta}:$$

$$P\text{-element: } p \in P_s \Delta$$

$$Q\text{-closure: } \theta \in \prod_{(y:r) \in \Delta} Q_r \Gamma$$

Scope-change as tensorial strength

$$\text{str}^{\mathbf{O}} : (\mathbf{O}P) \otimes \cdot \left(\text{var} \begin{array}{c} \mathbb{I} \\ \downarrow \\ A \end{array} \right) \rightarrow \mathbf{O} \left(P \otimes \cdot \left(\text{var} \begin{array}{c} \mathbb{I} \\ \downarrow \\ A \end{array} \right) \right)$$

identifying, e.g.:

$$\begin{aligned} [p[\text{weaken}], \theta]_{\Delta_1 \# \Delta_2} &= [p, \theta \circ \rho]_{\Delta_1} \\ [p[x', x'' \mapsto x]_{x \in \Delta}, \theta]_{\Delta} &= [p, \theta \# \theta]_{\Delta \# \Delta} \end{aligned}$$

MAST summary: semantic domain for syntax and semantics

MAST provides ($\mathbf{R} = (\text{sort}, \text{Bind})$ sorting system)

► Contexts

$$\text{Bind}_{\vdash} \ni \Gamma ::= [x_1 : s_1, \dots, x_n : s_n]$$

► Renamings

$$\text{Bind}_{\vdash}(\Gamma, \Delta) \ni \Gamma \vdash \rho : \Delta$$

► \mathbf{R} -structures:

$$\mathbf{PSh}(\text{sort} \times \text{Bind}_{\vdash}) \ni P : \text{sort} \times \text{Bind}_{\vdash}^{\text{op}} \rightarrow \mathbf{Set}$$

$$P_s \Gamma \ni p: \quad \text{sort } s \text{ element} \quad \text{with variables in } \Gamma$$

► Variables structure:

$$\mathbf{R}\text{-Struct} \ni \mathbb{I}_s \Gamma := \{x | (x : s) \in \Gamma\}$$

► substitution tensors:

$$\mathbf{R}\text{-Struct} \ni P \otimes Q, P \otimes \cdot \left(\text{var} \begin{array}{c} \mathbb{I} \\ \downarrow \\ A \end{array} \right)$$

$$(P \otimes Q)_s \Gamma \ni [p, \theta]_{\Delta}:$$

$$P\text{-element: } p \in P_s \Delta$$

$$Q\text{-closure: } \theta \in \prod_{(y:r) \in \Delta} Q_r \Gamma$$

Scope-change as tensorial strength

$$\text{str}^{\mathbf{O}} : (\mathbf{O}P) \otimes \cdot \left(\text{var} \begin{array}{c} \mathbb{I} \\ \downarrow \\ A \end{array} \right) \rightarrow \mathbf{O} \left(P \otimes \cdot \left(\text{var} \begin{array}{c} \mathbb{I} \\ \downarrow \\ A \end{array} \right) \right)$$

identifying, e.g.:

$$\begin{aligned} [p[\text{weaken}], \theta]_{\Delta_1 + \Delta_2} &= [p, \theta \circ \rho]_{\Delta_1} \\ [p[x', x'' \mapsto x]_{x \in \Delta}, \theta]_{\Delta} &= [p, \theta \# \theta]_{\Delta \# \Delta} \end{aligned}$$

Substitution monoids

$$\mathbf{M} \otimes \mathbf{M} \xrightarrow{-[-]} \mathbf{M} \xleftarrow{\text{var}} \mathbb{I}$$

What breaks the unitor?

Substitution tensor

$$(P \otimes Q)_s \Gamma := \int^{\Delta} P_s \Delta \times \prod_{(y:r) \in \Delta} Q_r \Gamma$$

for $s \notin \text{Bind}$, $Q = \mathbb{1}$, $\mathbb{I}_s \Delta = \emptyset$:

$$(\mathbb{I} \otimes Q)_s \Gamma := \int^{\Delta} \overbrace{\emptyset}^{\mathbb{I}_s \Delta} \times \prod_{(y:r) \in \Delta} Q_r \Gamma = \int^{\Delta} \emptyset = \textcolor{red}{\emptyset} \neq \mathbb{1} = Q_s \Gamma$$

MAST: semantic domain for **syntax**

Signature functors

$$\mathbf{O} \curvearrowright \mathbf{R}\text{-}\mathbf{Struct}$$

Scope-change as tensorial strength

$$\mathbf{str}^{\mathbf{O}} : (\mathbf{O}P) \otimes_{\bullet} \left(\mathbf{var} \begin{array}{c} \mathbb{I} \\ \downarrow \\ A \end{array} \right) \rightarrow \mathbf{O} \left(P \otimes_{\bullet} \left(\mathbf{var} \begin{array}{c} \mathbb{I} \\ \downarrow \\ A \end{array} \right) \right)$$

Example

$$\text{NB: } (\otimes_{\bullet}) : \mathbf{R}\text{-}\mathbf{Struct} \times (\mathbb{I}/\mathbf{R}\text{-}\mathbf{Struct}) \rightarrow \mathbf{R}\text{-}\mathbf{Struct}$$

Sequential fragment signature functor: $P \otimes_{\bullet} \left(\mathbf{var} \begin{array}{c} \mathbb{I} \\ \downarrow \\ A \end{array} \right) := P \otimes A$

$$(\mathbf{Seq} X)_{\text{comp } B} \Gamma := \coprod_{A \in \text{Type}} \left((\mathbf{let } x : A = _ \mathbf{in } _) : (X_{\text{comp } A} \Gamma \times X_{\text{comp } B} (\Gamma, x : A)) \right)$$

$$(\mathbf{Seq} X)_A \Gamma := \emptyset$$

$$\mathbf{str}^{\mathbf{Seq}} [\mathbf{let } x : A = (p \in P_{\text{comp } A} \Delta) \mathbf{in } (q \in P_{\text{comp } B} (\Delta, x : A)), \theta]_{\Delta}$$

Takeaway (modularity)

$$:= (\mathbf{let } x : A = [p, \theta]_{\Delta} \mathbf{in } [q, (\theta, x : \mathbf{var } x)]_{\Delta, x : A})$$

Each syntactic construct defines its own binding, renaming, and substitution structure

Signatrue combinators

[cf. SOAS]

- ▶ sums & products of signature functors
- ▶ scope extension ($\Gamma \triangleright$)
- ▶ sort extension $\varphi_s : \mathbf{PSh} \text{ Bind}_\perp \rightarrow \mathbf{PSh} (\text{sort} \times \text{Bind}_\perp)$
- ▶ sort application $(@s) : \mathbf{PSh} (\text{sort} \times \text{Bind}_\perp) \rightarrow \mathbf{PSh} \text{ Bind}_\perp$

Example (Binding signatures [Actzel'78])

NB

$\text{Seq} \cong$

$(\text{Seq } X)_{\text{comp } B} \Gamma :=$

$$\coprod_{A \in \text{Type}} \left(\left(\text{let } x : A = _ \text{ in } _ \right) : \varphi_{\text{comp } B} ((@ \text{comp } A) \times ([x : A] \triangleright)) \right)$$

$$\coprod_{A \in \text{Type}} \left(\left(\text{let } x : A = _ \text{ in } _ \right) : (X_{\text{comp } A} \Gamma \times X_{\text{comp } B} (\Gamma, x : A)) \right)$$

$(\text{Seq } X)_A \Gamma := \emptyset$

MAST: semantic domain for **syntax**

Abstract syntax: inductive representation

Every initial algebra:

$$\mathcal{S}^0 \mathbf{H} := \mu X. (\mathbf{O}X) \sqcup \mathbb{I} \sqcup \mathbf{H} \otimes X$$

Supports standard definitions:

$$\begin{array}{c} \mathbf{H} \\ \downarrow \text{?} - [-] \\ \mathcal{S}\mathbf{H} \otimes \mathcal{S}\mathbf{H} \xrightarrow{-[-]} \mathcal{S}\mathbf{H} \xleftarrow[\text{var}]{\mathbb{I}} \mathbb{I} \\ \uparrow [-] \\ \mathbf{O}(\mathcal{S}\mathbf{H}) \end{array}$$

Independently of concrete representation, e.g.,:

- ▶ De-Bruijn
- ▶ Locally nameless
- ▶ Graphical
- ▶ Nominal
- ▶ Co-de Bruijn

Example

$\mathbf{M} = (\mathcal{C}, \mathbf{T}, \text{return}, \gg=, \llbracket - \rrbracket)$:

- ▶ \mathcal{C} : Cartesian category with chosen finite products
- ▶ $(\mathbf{T}, \text{return}, \gg=)$ strong monad over \mathcal{C}
- ▶ $\llbracket - \rrbracket : \text{Type} \rightarrow \mathcal{C}$ type interpretation

induces:

- ▶ A CBV-structure: $\text{CBV-Struct} \ni \mathbf{M}_s \Gamma := \mathcal{C}(\llbracket \Gamma \rrbracket, \llbracket s \rrbracket)$
- ▶ Standard interpretation of contexts, computations, renaming:

$$\begin{aligned} \mathcal{C} \ni \llbracket \Gamma \rrbracket &:= \prod_{(x:A) \in \Gamma} \llbracket A \rrbracket & \mathcal{C} \ni \llbracket \text{comp } A \rrbracket &:= \mathbf{T} \llbracket A \rrbracket \\ \llbracket \rho \rrbracket : \llbracket \Gamma \rrbracket &\xrightarrow{(\pi_{x[\rho]} : \llbracket \Gamma \rrbracket \rightarrow \llbracket A \rrbracket)_{(x:A) \in \Delta}} \prod_{(x:A) \in \Delta} \llbracket A \rrbracket = \llbracket \Delta \rrbracket \end{aligned}$$

MAST: common structure for substitution

Syntactic substitution monoid

$$\mathcal{S}^0\mathbf{H} \otimes \mathcal{S}^0\mathbf{H} \xrightarrow{-[-]} \mathcal{S}^0\mathbf{H} \xleftarrow{\text{var}} \mathbb{I}$$

Monoid axioms amount to syntactic substitution lemma

Example

Semantic substitution monoid:

$$\mathbf{M} \otimes \mathbf{M} \xrightarrow{-[-]} \mathbf{M} \xleftarrow{\text{var}} \mathbb{I}$$

- Substitution via composition:

$$\left(\llbracket \Delta \rrbracket \xrightarrow{f} \llbracket s \rrbracket \right) \left[\llbracket \Gamma \rrbracket \xrightarrow{\theta} \llbracket \Delta \rrbracket \right] : \llbracket \Gamma \rrbracket \xrightarrow{\theta} \llbracket \Delta \rrbracket \xrightarrow{f} \llbracket s \rrbracket$$

- Variables:

(1st-class sorts only)

$$\text{var} : \left((x : A) \in \Gamma \mapsto \left(\llbracket \Gamma \rrbracket \xrightarrow{\pi_x} \llbracket A \rrbracket \right) \right)$$

MAST: compatibility

Substitution-compatible algebra

$\llbracket - \rrbracket : \mathbf{OM} \rightarrow \mathbf{M}$:

$$\begin{array}{ccc}
 & \xrightarrow{\text{str}} & \underline{\mathbf{O}}(\underline{\mathbf{M}} \otimes \underline{\mathbf{M}}) \\
 (\underline{\mathbf{OM}}) \otimes \cdot \text{var}^{\mathbf{M}} & \xrightarrow{\text{compatibility}} & \underline{\mathbf{O}}(-\llbracket - \rrbracket_{\mathbf{M}}) \\
 \llbracket - \rrbracket \otimes \cdot \text{id} \searrow & = & \searrow \underline{\mathbf{OM}} \\
 & & \llbracket - \rrbracket \\
 \underline{\mathbf{M}} \otimes \cdot \text{var}^{\mathbf{M}} & \xrightarrow{-\llbracket - \rrbracket_{\mathbf{M}}} & \underline{\mathbf{M}}
 \end{array}$$

Example (Seq-algebra)

$\left[\begin{array}{l} \text{let } x : A = (\llbracket \Gamma \rrbracket \xrightarrow{f} T \llbracket A \rrbracket) \\ \text{in } (\llbracket \Gamma \rrbracket \times \llbracket A \rrbracket \xrightarrow{g} T \llbracket B \rrbracket) \end{array} \right] :$

$$\llbracket \Gamma \rrbracket \xrightarrow{(\text{id}, f)} \llbracket \Gamma \rrbracket \times T \llbracket A \rrbracket \xrightarrow{\not\approx^g} T \llbracket B \rrbracket$$

Takeaway

Equip each semantic interpretation with its compatibility proof

Example (Seq-compatibility)

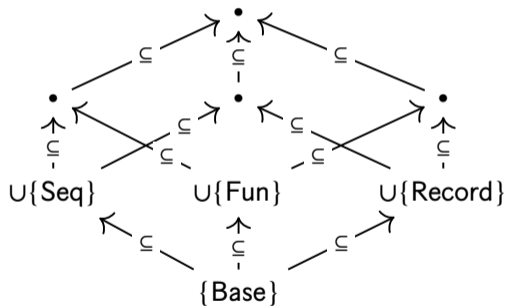
Compatibility:

$$\begin{array}{ccccc}
 \llbracket \Gamma \rrbracket & \xrightarrow{(\text{id}, (f \circ \theta))} & \llbracket \Gamma \rrbracket \times T \llbracket A \rrbracket & \xrightarrow{\not\approx^{(g \circ (\theta \times \text{id}))}} & T \llbracket B \rrbracket \\
 \theta \downarrow & \text{products} & \downarrow \theta \times \text{id} & \text{strong monad laws} & \\
 \llbracket \Delta \rrbracket & \xrightarrow{(\text{id}, f)} & \llbracket \Delta \rrbracket \times T \llbracket A \rrbracket & \xrightarrow{\not\approx^g} & T \llbracket B \rrbracket \\
 & & & = &
 \end{array}$$

MAST: modularity and scalability

Substitution **O**-monoid

Substitution monoid with compatible **O**-algebra structure



Want more?

In the paper:

- ▶ All the details
- ▶ A CBV case-study (128 substitution lemmata)



Talk structure

- ▶ Contribution
- ▶ Substitution monoids
- ▶ MAST in detail
- ▶ **WIP**

SMTLIB Foreign Function Interface (FFI)

Implementation

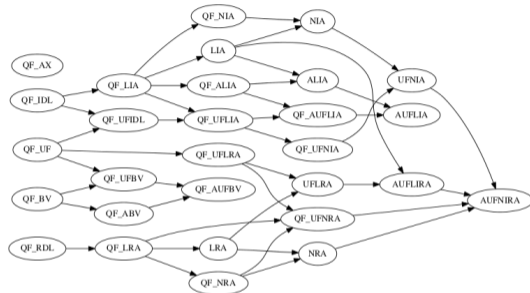
Idris 2 [Brady'21] implementation
of computational fragment
[cf. Fiore and Szamozvancev'22]

SMTLIB query language

- ▶ S-expressions
- ▶ 29 theories
- ▶ multiple syntax extensions

FFI

- ▶ Intrinsically-typed well-scoped FFI with holes
- ▶ Modular serialisation
- ▶ Modular well-scoped parsing
- ▶ Modular type-inference



[Greg Brown'25]

Non-skew structure with actions

(time permitting on board)

[cf. Fiore and Turi'01]

Modular Abstract Syntax Trees (MAST)

- ▶ SOAS $\xrightarrow{\text{generalise}}$ 2nd-class sorts
Using **skew** bicategories/monoidal categories, and:
 - ▶ Kleisli bicategories [Gambino, Fiore, Hyland, and Winskel'19]
 - ▶ Familial theory of SOAS [Fiore and Szamozvancev'25]
- ▶ MAST tutorial
- ▶ Case-study: CBV semantics á la carte (128 substitution lemmata)

WIP

- ▶ Idris 2 implementation of computational fragment [cf. Fiore and Szamozvancev'22]
Case-study: intrinsically-typed FFI-binding with holes for SMTLIB (29 theories)
- ▶ Replace skew monoidal structure and monoids with
monoidal structure and actions

[cf. Fiore and Turi'01]