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

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

Paper: Slides:

Theoretical Computer Science Seminar
24 October 2025
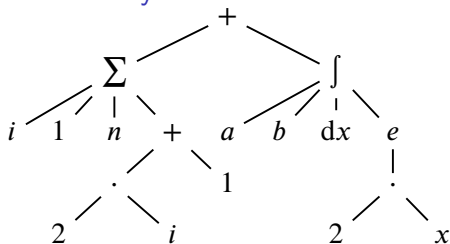Department of Computer Science, University of Birmingham, England

$$\left(\sum_{i=1}^{n}(2i+1)\right)+\int_{a}^{b}e^{ax}\mathrm{d}x$$
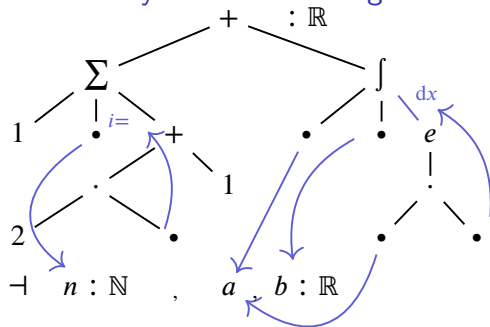
Concrete syntax

"(", "$\sum$", "₋", "{", "i", "=", "1", "}", "{", "n", "(", "2", "i", "+", "1", ")", "}", ...

Abstract syntax

Abstract syntax with binding

Call-by-Value λ-calculus

$$A, B, C ::= \qquad \text{type}$$
$$\quad \beta \qquad\qquad \text{base}$$
$$\mid\ A \to B \qquad \text{function}$$
$$\mid\ (\!(C_i : A_i \mid i \in I)\!) \quad \text{record } (I \text{ finite})$$
$$\mid\ \{\!|C_i : A_i \mid i \in I|\!\} \quad \text{variant } (I \text{ finite})$$
$$\vdots$$

$$V, W ::= \qquad \text{value}$$
$$\quad x \qquad\qquad \text{variable}$$
$$\mid\ \lambda x : A.M \qquad \text{function abst.}$$
$$\mid\ (\!(C_i : V_i \mid i \in I)\!) \quad \text{record c'tor}$$
$$\mid\ A.C_i\ V \qquad \text{variant c'tor}$$
$$\vdots$$

$$M, N, K, L ::= \qquad\qquad\qquad\qquad\qquad \text{term}$$
$$\quad \mathsf{val}\ V \qquad\qquad\qquad\qquad\qquad\quad \text{value}$$
$$\mid\ \mathbf{let}\ x_1 = M_1; \dots; x_n = M_n\ \mathbf{in}\ N \qquad \text{sequencing}$$
$$\mid\ M @ N \qquad\qquad\qquad\qquad\qquad \text{function application}$$
$$\mid\ (\!(C_1 : M_1, \dots, C_n : M_n)\!) \qquad\qquad \text{record constructor}$$
$$\mid\ \mathbf{case}\ M\ \mathbf{of}\ (\!(C_1 x_1, \dots, C_n x_n)\!) \Rightarrow N \quad \text{record pattern match}$$
$$\mid\ A.C_i\ M \qquad\qquad\qquad\qquad\qquad \text{variant constructor}$$
$$\mid\ \mathbf{case}\ M\ \mathbf{of}\ \{C_i x_i \Rightarrow M_i \mid i \in I\}\ N \quad \text{variant pattern match}$$
$$\vdots$$

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
$\vdots$			
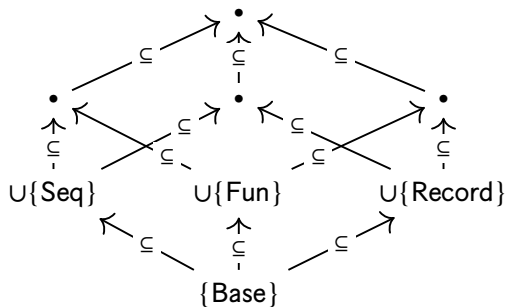
Dream

Iterative semantic development
- ▶ Add syntax
- ▶ Add semantics
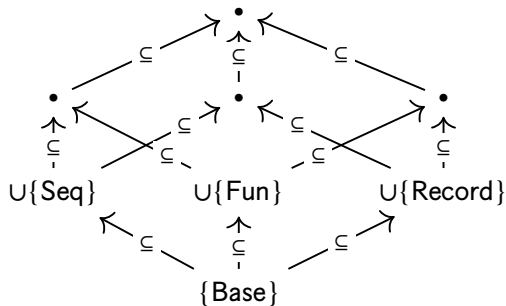

- ▶ Profit!

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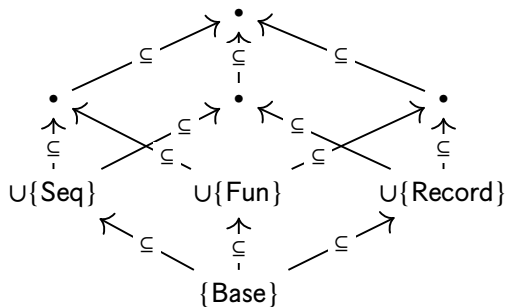
Iterative semantic development

- ▶ 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:

$$[\![M \, [\theta]]\!] = [\![M]\!] \circ [\![\theta]\!]$$

Lemma (compositionality)

Composite semantics is independent of component syntax:

$$[\![C[M]]\!] = \mathrm{plug}([\![C[-]]\!] , [\![M]\!])$$

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:

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

Proof.

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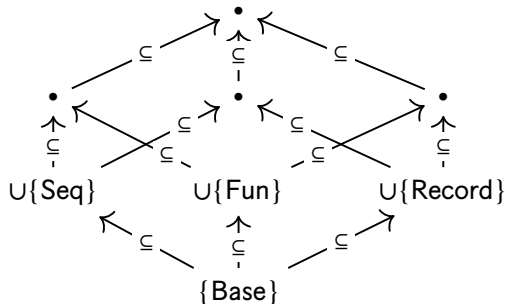
Dream

It would be nice if tedious bits were…
… free

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   [Fiore and Hamana'13]
  - ▶ mechanisation   [Crole'11, Allais et al.'18,
                                 Fiore and Szamozvancev'22]
  - ▶ substructurality   [Fiore and Ranchod'25]
- ▶ CBN works smoothly. Doesn't cover CBV.
  Technical reasons later:
  - ▶ Substitute **in**: values and terms
  - ▶ Substitute for variables: values only

Slogan

for substitution: values are 1st-class   but   terms are 2nd-class

[cf. Levy's CBPV, '04]

Sorting signature $\mathbf{R}$

- set sort

partitioned into

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

Example (CBV sorting signature)

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

Example (CBPV sorting signature)

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

Core contribution

classical theory (SOAS)   this work (MAST)

$\mathbf{PSh}\,(\text{sort} \times \text{sort}_\vdash), \otimes$ $\overset{\text{generalise}}{\rightsquigarrow}$ $\mathbf{PSh}\,(\text{sort} \times \text{Bind}_\vdash), \otimes$

monoidal product   right-unital associative
                                           skew monoidal product

Skew monoidal heterogeneous tensor

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

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

$$\mathbb{I} \otimes Q \xrightarrow{\ell} Q \qquad \text{(non-invertible!)} \qquad (\mathbb{I} \otimes \mathbb{1})_s = \emptyset \not\cong \mathbb{1}_s (s \notin \text{Bind})$$

Contribution

Modular Abstract Syntax Trees (MAST)

- SOAS $\overset{\text{generalise}}{\leadsto}$ 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

Thm (representation)

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

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

Plugging holes/metavariable substitution

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

Capstone: semantics

Key propaganda

compositional, binding-respecting denotational semantics
amounts to
substitution $\mathbf{O}$-monoid:

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

The denotational semantics for terms with holes in $\mathbf{H}$ is the unique substitution $\mathbf{O}$-monoid homomorphism over $\mathbf{H}$:

$$\left(\$\mathbf{H}, -[-], \mathsf{var}, \llbracket - \rrbracket, ?-[\mathsf{id}] \right) \xrightarrow{\ \llbracket - \rrbracket\ } (\mathbf{M}, -[-], \mathsf{var}, \llbracket - \rrbracket, \mathsf{menv}) \qquad (\mathbf{H} \xrightarrow{\ \mathsf{menv}\ } \mathbf{M})$$

Meta-theory in one line

Lemma (substitution)

Syntactic substitution corresponds to semantic composition:

$$[\![M\,[\theta]]\!] = \qquad\qquad\qquad [\![M]\!] \circ [\![\theta]\!]$$

Lemma (compositionality)

Composite semantics is independent of component syntax:

$$[\![C[M]]\!] =$$
$$\mathrm{plug}([\![C[-]]\!]\,,\,[\![M]\!])$$

Lemma (substitution)

Syntactic substitution corresponds to semantic composition:

$$\overset{\substack{\text{substitution monoid homomorphism} \\ \downarrow}}{[\![M\,[\theta]]\!] \;=\; [\![-[-]\,[M,\theta]]\!] \;=\; -[-]\,\big[[\![M]\!]\,,[\![\theta]\!]\big] \;:=\; [\![M]\!] \circ [\![\theta]\!]}$$

Lemma (compositionality)

Composite semantics is independent of component syntax:

$$[\![C[M]]\!] \;=\;$$
$$\operatorname{plug}([\![C[-]]\!]\,,[\![M]\!])$$

Lemma (substitution)

Syntactic substitution corresponds to semantic composition:

$$\underset{\underset{\downarrow}{\text{substitution monoid homomorphism}}}{} $$

$$[\![M\,[\theta]]\!] = [\![-[-]\,[M,\theta]]\!] = -[-]\,\big[[\![M]\!]\,,[\![\theta]\!]\big] := [\![M]\!] \circ [\![\theta]\!]$$

Lemma (compositionality)

Composite semantics is independent of component syntax:

$$[\![C[M]]\!] = [\![C[?m] \gg\!\!= (m \mapsto M)]\!] = [\![C[?m]]\!] \gg\!\!= (m \mapsto [\![M]\!])$$

$$=: \text{plug}([\![C[-]]\!]\,,[\![M]\!])\underset{\underset{\substack{homomorphic \\ extension}}{\gg\!\!=\ is}}{\uparrow}$$

Talk structure

- Contribution
- Substitution monoids
- MAST in detail
- WIP

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

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

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

- $\mathbf{R}$-structures:
$$\mathbf{PSh}(\text{sort} \times \text{Bind}_\vdash) \ni P : \text{sort} \times \text{Bind}_\vdash^{\text{op}} \to \mathbf{Set}$$
$$P_s\Gamma \ni p: \quad \text{sort } s \text{ element} \quad \text{with variables in } \Gamma$$

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

- substitution tensors:
$$\mathbf{R\text{-}Struct} \ni P \otimes Q, P \otimes_{\bullet} \left(\text{var} \begin{smallmatrix} \mathbb{I} \\ \downarrow \\ A \end{smallmatrix} \right)$$
$(P \otimes Q)_s\Gamma \ni [p, \theta]_\Delta$:
$$P\text{-element:} \quad p \in P_s\Delta$$
$$Q\text{-closure}: \quad \theta \in \prod_{(y:r)\in\Delta} Q_r\Gamma$$

identifying, e.g.:
$$[p[\text{weaken}], \theta]_{\Delta_1 \uplus \Delta_2} = [p, \theta \circ \rho]_{\Delta_1}$$
$$\left[p[x', x'' \mapsto x]_{x \in \Delta}, \theta \right]_\Delta = [p, \theta \uplus \theta]_{\Delta \uplus \Delta}$$

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

- Contexts $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\text{Bind}_\vdash \ni \Gamma ::= [x_1 : s_1, \ldots, x_n : s_n]$
- Renamings $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\text{Bind}_\vdash(\Gamma, \Delta) \ni \Gamma \vdash \rho : \Delta$
- $\mathbf{R}$-structures: $\qquad\qquad$ $\mathbf{PSh}(\text{sort} \times \text{Bind}_\vdash) \ni P : \text{sort} \times \text{Bind}_\vdash^{\text{op}} \to \mathbf{Set}$

  $P_s\Gamma \ni p$:   sort s element   with variables in Γ
- Variables structure: $\qquad\qquad\qquad\qquad$ $\mathbf{R\text{-}Struct} \ni \mathbb{I}_s\Gamma := \{x | (x : s) \in \Gamma\}$
- substitution tensors: $\qquad\qquad\qquad\qquad$ $\mathbf{R\text{-}Struct} \ni P \otimes Q, P \otimes_\bullet \left(\text{var} \underset{A}{\overset{\mathbb{I}}{\downarrow}} \right)$

  $(P \otimes Q)_s\Gamma \ni [p, \theta]_\Delta$: $\qquad$ P-element: $p \in P_s\Delta$

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

Scope-change as tensorial strength $\qquad$ identifying, e.g.:

$\text{str}^{\mathbf{O}} : (\mathbf{O}P) \otimes_\bullet \left(\text{var} \underset{A}{\overset{\mathbb{I}}{\downarrow}} \right) \to \mathbf{O}\left(P \otimes_\bullet \left(\text{var} \underset{A}{\overset{\mathbb{I}}{\downarrow}} \right) \right)$ $\qquad$ $[p[\text{weaken}], \theta]_{\Delta_1 + \Delta_2} = [p, \theta \circ \rho]_{\Delta_1}$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\left[p[x', x'' \mapsto x]_{x \in \Delta}, \theta \right]_\Delta = [p, \theta + \theta]_{\Delta + \Delta}$

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

- Contexts $\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\text{Bind}_\vdash \ni \Gamma ::= [x_1 : s_1, \ldots, x_n : s_n]$
- Renamings $\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\text{Bind}_\vdash(\Gamma, \Delta) \ni \Gamma \vdash \rho : \Delta$
- $\mathbf{R}$-structures: $\qquad\qquad$ $\mathbf{PSh}(\text{sort} \times \text{Bind}_\vdash) \ni P : \text{sort} \times \text{Bind}_\vdash^{\text{op}} \to \mathbf{Set}$

  $P_s\Gamma \ni p$: $\quad$ sort s element $\quad$ with variables in Γ
- Variables structure: $\qquad\qquad\qquad$ $\mathbf{R}\text{-Struct} \ni \mathbb{I}_s\Gamma := \{x | (x : s) \in \Gamma\}$
- substitution tensors: $\qquad\qquad\qquad$ $\mathbf{R}\text{-Struct} \ni P \otimes Q, P\otimes_\bullet\left(\text{var} \underset{A}{\overset{\mathbb{I}}{\downarrow}}\right)$

  $(P \otimes Q)_s\Gamma \ni [p, \theta]_\Delta$: $\qquad$ P-element: $\quad p \in P_s\Delta$

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

Scope-change as tensorial strength

$\text{str}^{\mathbf{O}} : (\mathbf{O}P)\otimes_\bullet\left(\text{var} \underset{A}{\overset{\mathbb{I}}{\downarrow}}\right) \to \mathbf{O}\left(P\otimes_\bullet\left(\text{var} \underset{A}{\overset{\mathbb{I}}{\downarrow}}\right)\right)$

Substitution monoids

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

identifying, e.g.:

$[p[\text{weaken}], \theta]_{\Delta_1 + \Delta_2} = [p, \theta \circ \rho]_{\Delta_1}$

$\left[p[x', x'' \mapsto x]_{x\in\Delta}, \theta\right]_\Delta = [p, \theta + \theta]_{\Delta + \Delta}$

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

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

Signature functors

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

Scope-change as tensorial strength

$$\mathsf{str}^{\mathbf{O}} : (\mathbf{O}P) \otimes_{\bullet} \left(\mathsf{var} \begin{smallmatrix} \mathbb{I} \\ \downarrow \\ A \end{smallmatrix} \right) \to \mathbf{O} \left(P \otimes_{\bullet} \left(\mathsf{var} \begin{smallmatrix} \mathbb{I} \\ \downarrow \\ A \end{smallmatrix} \right) \right)$$

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

Example

Sequential fragment signature functor:

$$P \otimes_{\bullet} \left(\mathsf{var} \begin{smallmatrix} \mathbb{I} \\ \downarrow \\ A \end{smallmatrix} \right) \coloneqq P \otimes A$$

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

$$(\mathsf{Seq}\, X)_A \Gamma \coloneqq \emptyset$$

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

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

Takeaway (modularity)

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

Signatrue combinators [cf. SOAS]

- sums & products of signature functors
- scope extension ($\Gamma \rhd$)
- sort extension $\underset{s}{\hooklefthalfright}$: $\mathbf{PSh}\,\mathsf{Bind}_\vdash \to \mathbf{PSh}\,(\mathsf{sort} \times \mathsf{Bind}_\vdash)$
- sort application ($@\,s$) : $\mathbf{PSh}\,(\mathsf{sort} \times \mathsf{Bind}_\vdash) \to \mathbf{PSh}\,\mathsf{Bind}_\vdash$

Example (Binding signatures [Actzel'78])

$\mathsf{Seq} \cong$
$$\coprod\nolimits_{A \in \mathsf{Type}} \begin{pmatrix} (\mathbf{let}\ x : A = \_\ \mathbf{in}\ \_) : \\ \underset{\mathsf{comp}\,B}{\hooklefthalfright} ((@\,\mathsf{comp}\,A) \times ([x : A]\rhd)) \end{pmatrix}$$

NB
$(\mathsf{Seq}\,X)_{\mathsf{comp}\,B}\Gamma :=$
$$\coprod_{A \in \mathsf{Type}} \begin{pmatrix} (\mathbf{let}\ x : A = \_\ \mathbf{in}\ \_) : \\ \left(X_{\mathsf{comp}\,A}\Gamma \times X_{\mathsf{comp}\,B}\,(\Gamma, x : A)\right) \end{pmatrix}$$
$(\mathsf{Seq}\,X)_A\Gamma := \emptyset$

Abstract syntax: inductive representation

Every initial algebra:

$$\$^{\mathbf{O}}\mathbf{H} \coloneqq \boldsymbol{\mu} X.(\mathbf{O}X) \amalg \mathbb{I} \amalg \mathbf{H} \otimes X$$

Supports standard definitions:

$$
\$\mathbf{H} \otimes \$\mathbf{H} \xrightarrow{\;\;-[-]\;\;}
\begin{array}{c}
\mathbf{H} \\
\downarrow\, ?-[-] \\
\$\mathbf{H} \xleftarrow[\mathsf{var}]{} \mathbb{I} \\
[\![-]\!] \uparrow \\
\mathbf{O}(\$\mathbf{H})
\end{array}
$$

Independently of concrete representation, e.g.,:

- ▶ De-Bruijn
- ▶ Nominal

- ▶ Locally nameless
- ▶ Co-de Bruijn

- ▶ Graphical

Example

$\mathbf{M} = \big(C, \mathrm{T}, \mathrm{return}, \ggeq, [\![-]\!] \big):$

- ▶ C: Cartesian category with chosen finite products
- ▶ $(\mathrm{T}, \mathrm{return}, \ggeq)$ strong monad over C
- ▶ $[\![-]\!] : \mathsf{Type} \to C$ type interpretation

induces:

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

$$C \ni [\![\Gamma]\!] := \prod_{(x:A) \in \Gamma} [\![A]\!] \qquad C \ni [\![\mathsf{comp}\,A]\!] := \mathrm{T}\,[\![A]\!]$$

$$[\![\rho]\!] \,:\, [\![\Gamma]\!] \xrightarrow{\ (\pi_{x[\rho]} : [\![\Gamma]\!] \to [\![A]\!])_{(x:A) \in \Delta}\ } \prod_{(x:A) \in \Delta} [\![A]\!] = [\![\Delta]\!]$$

MAST: common structure for substitution

Syntactic substitution monoid

$$\mathbb{S}^{\mathbf{O}}\mathbf{H} \otimes \mathbb{S}^{\mathbf{O}}\mathbf{H} \xrightarrow{-[-]} \mathbb{S}^{\mathbf{O}}\mathbf{H} \xleftarrow{\mathsf{var}} \mathbb{I}$$

Monoid axioms amount to syntactic substitution lemma

Example

Semantic substitution monoid:
$$\mathbf{M} \otimes \mathbf{M} \xrightarrow{-[-]} \mathbf{M} \xleftarrow{\mathsf{var}} \mathbb{I}$$

▶ Substitution via composition:

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

▶ Variables:
$$\mathsf{var} : \left((x:A) \in \Gamma \mapsto \left([\![\Gamma]\!] \xrightarrow{\pi_x} [\![A]\!] \right) \right)$$
(1st-class sorts only)

Substitution-compatible algebra

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

$$
\begin{array}{ccc}
& \xrightarrow{\mathrm{str}} \underline{\mathbf{O}}(\underline{\mathbf{M}} \otimes \underline{\mathbf{M}}) & \xrightarrow{\underline{\mathbf{O}}(-[-]_{\mathbf{M}})} \\
(\underline{\mathbf{OM}}) \otimes_\bullet \mathrm{var}^{\mathbf{M}} & \text{compatibility} & \underline{\mathbf{OM}} \\
\llbracket - \rrbracket \otimes_\bullet \mathrm{id} \downarrow & = & \downarrow \llbracket - \rrbracket \\
\underline{\mathbf{M}} \otimes_\bullet \mathrm{var}^{\mathbf{M}} & \xrightarrow{-[-]_{\mathbf{M}}} & \underline{\mathbf{M}}
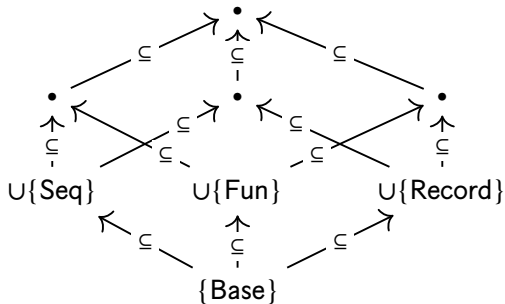\end{array}
$$

Example (Seq-compatibility)

Compatibility:

$$
\begin{array}{ccc}
\llbracket \Gamma \rrbracket & \xrightarrow{(\mathrm{id},(f\circ\theta))} & \llbracket \Gamma \rrbracket \times \mathrm{T}\, \llbracket A \rrbracket \\
\theta \downarrow & \overset{\text{products}}{=} & \downarrow \theta\times\mathrm{id} \\
\llbracket \Delta \rrbracket & \xrightarrow{(\mathrm{id},f)} & \llbracket \Delta \rrbracket \times \mathrm{T}\, \llbracket A \rrbracket
\end{array}
$$

$\xrightarrow{\,\succcurlyeq (g\circ(\theta\times\mathrm{id}))\,}$ strong monad laws $=$ $\xrightarrow{\,\succcurlyeq g\,}$ $\mathrm{T}\, \llbracket B \rrbracket$

Example (Seq-algebra)

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

$$
\llbracket \Gamma \rrbracket \xrightarrow{(\mathrm{id},f)} \llbracket \Gamma \rrbracket \times \mathrm{T}\, \llbracket A \rrbracket \xrightarrow{\succcurlyeq g} \mathrm{T}\, \llbracket B \rrbracket
$$

Takeaway

Equip each semantic interpretation with its compatibility proof

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

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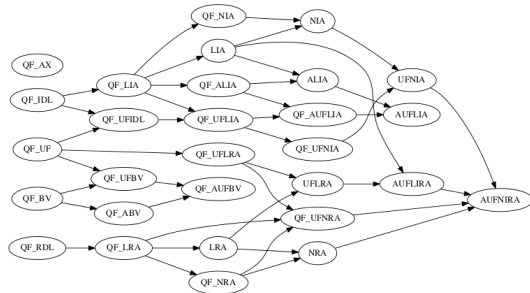
Idris 2 [Brady'21] implementation
of computational fragment
[cf. Fiore and Szamozvancev'22]

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]

(time permitting on board)   [cf. Fiore and Turi'01]

Contribution

Modular Abstract Syntax Trees (MAST)

- SOAS $\overset{\text{generalise}}{\rightsquigarrow}$ 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]