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:

Seminar za temelje matematike in teoretično računalništvo
(Foundations of Mathematics and Computer Science Seminar)
18 September 2025
Faculty of Mathematics and Physics, University of Ljubljana, Slovenia

THE UNIVERSITY of EDINBURGH informatics lfcs Laboratory for Foundations of Computer Science BayesCentre

Funded by: THE ROYAL SOCIETY Advanced Research + Invention Agency ARIA erc European Research Council Established by the European Commission FWF Austrian Science Fund

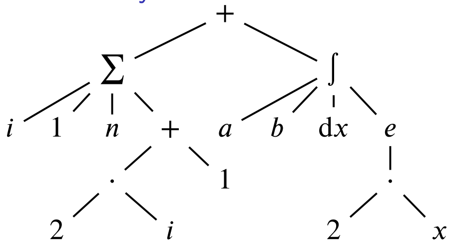
Syntax representation

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

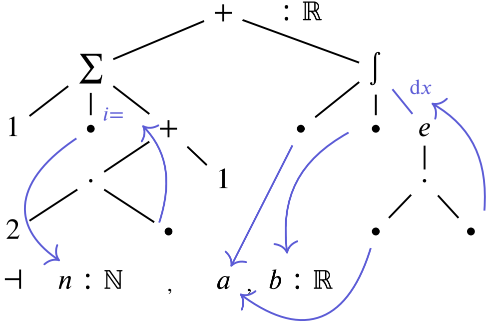
Concrete syntax

“(”, “$\sum$”, “_”, “{”, “i”, “=”, “1”, “}”, “{”, “n”, “(”, “2”, “i”, “+”, “1”, “)”, “}”, …

Abstract syntax



Abstract syntax with binding



Call-by-Value λ-calculus

$$
\begin{array}{rll}
A, B, C ::= & & \text{type} \\
& \beta & \text{base} \\
| & A \to B & \text{function} \\
| & (\!| C_i : A_i \,|\, i \in I |\!) & \text{record } (I \text{ finite}) \\
| & \{\!| C_i : A_i \,|\, i \in I |\!\} & \text{variant } (I \text{ finite}) \\
\vdots & &
\end{array}
$$

$$
\begin{array}{rll}
V, W ::= & & \text{value} \\
& x & \text{variable} \\
| & \lambda x : A.M & \text{function abst.} \\
| & (C_i : V_i \,|\, i \in I) & \text{record c'tor} \\
| & A.C_i\ V & \text{variant c'tor} \\
\vdots & &
\end{array}
$$

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

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

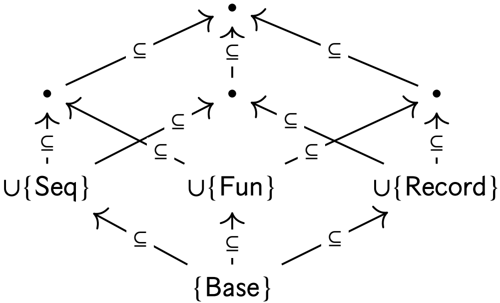
fragment	syntactic constructs	types	semantics
base	returning a value: val		strong monad over a Cartesian category
sequential	sequencing: **let**		
functions	abst., app. $(\lambda x.:A), (@)$	function $(\to)$	Kleisli exponentials
variants	c'tors, pattern match $A.C_i -$, $\textbf{case} - \textbf{of} \{C_i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

⋮

Dream

Iterative semantic development

- Add syntax
- Add semantics

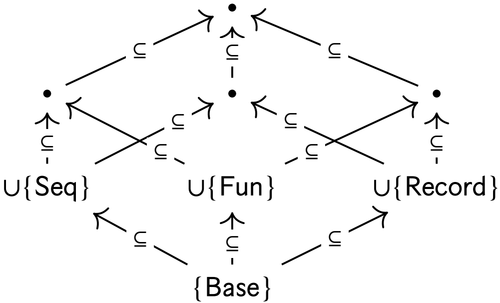


- Profit!

Dream vs. Reality

Iterative semantic development

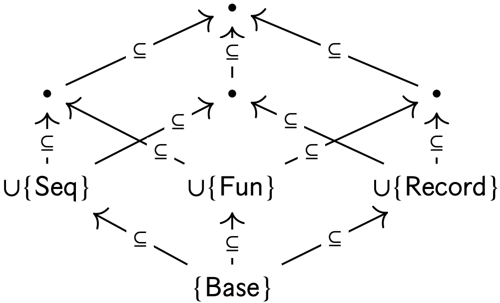
- Add syntax
- Add semantics
- Develop meta-theory:
  - Substitution lemma
  - Compositionality
  - Soundness
  - Adequacy
- Profit!



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:

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

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:

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

Proof.

Presupposes a syntactic substitution lemma. Typically several inductions over all constructs. $\square$

Lemma (compositionality)

Composite semantics is independent of component syntax:

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

Proof.

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

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

$$
\begin{tikzcd}
 & \bullet & \\
\bullet \arrow[ur, hook, "\subseteq"] & \bullet \arrow[u, hook, "\subseteq"] & \bullet \arrow[ul, hook, "\subseteq"'] \\
\cup\{\mathsf{Seq}\} \arrow[u, hook, "\subseteq"] \arrow[ur, hook, "\subseteq"] & \cup\{\mathsf{Fun}\} \arrow[ul, hook, "\subseteq"] \arrow[ur, hook, "\subseteq"] & \cup\{\mathsf{Record}\} \arrow[u, hook, "\subseteq"] \arrow[ul, hook, "\subseteq"] \\
 & \{\mathsf{Base}\} \arrow[ul, hook, "\subseteq"] \arrow[u, hook, "\subseteq"] \arrow[ur, hook, "\subseteq"'] &
\end{tikzcd}
$$

Expression problem [Reynolds'75, Cook'90, Krishnamurthi, Felleisen and Friedman'98, Wadler'98]

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]

SOAS: Second-Order Abstract Syntax [Fiore, Plotkin, and Turi '99]

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

Example (CBV sorting signature)

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

Example (CBPV sorting signature)

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

Core contribution

classical theory (SOAS)	generalise	this work (MAST)
$\mathbf{PSh}(\mathsf{sort} \times \mathsf{sort}_{\vdash}), \otimes$ monoidal product	$\rightsquigarrow$	$\mathbf{PSh}(\mathsf{sort} \times \mathsf{Bind}_{\vdash}), \otimes$ right-unital associative **skew** monoidal product

Skew monoidal heterogeneous tensor

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

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

$$\mathbb{I} \otimes Q \xrightarrow{\ell} Q \qquad \text{(non-invertible!)} \qquad (\mathbb{I} \otimes \mathbb{1})_s = \emptyset \ncong \mathbb{1}_s (s \notin \mathsf{Bind})$$

Contribution

Modular Abstract Syntax Trees (MAST)

- SOAS $\overset{\text{generalise}}{\rightsquigarrow}$ 2[nd]-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 $\mathbf{H}$:

$$
\begin{tikzcd}
 & \mathbf{H} \arrow[d, "?{-}[{-}]"] & \\
\$\mathbf{H} \otimes \$\mathbf{H} \arrow[r, "{-}[{-}]"] & \$\mathbf{H} & \mathbb{I} \arrow[l, "\mathsf{var}"'] \\
 & \mathbf{O}(\$\mathbf{H}) \arrow[u, "[\![-]\!]"] &
\end{tikzcd}
$$

Plugging holes/metavariable substitution

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

Capstone: semantics

Key propaganda

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

$$
\begin{array}{ccccc}
\mathbf{M} \otimes \mathbf{M} & \xrightarrow{-[-]} & \mathbf{M} & \xleftarrow[\mathsf{var}]{} & \mathbb{I} \\
 & & [\![-]\!] \uparrow & & \\
 & & \mathbf{OM} & &
\end{array}
$$

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

$$
\big(\$\mathbf{H}, -[-], \mathsf{var}, [\![-]\!], ?{-}[\mathsf{id}]\big) \xrightarrow{[\![-]\!]} (\mathbf{M}, -[-], \mathsf{var}, [\![-]\!], \mathsf{menv}) \qquad (\mathbf{H} \xrightarrow{\mathsf{menv}} \mathbf{M})
$$

Lemma (substitution)

Syntactic substitution corresponds to semantic composition:

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

Lemma (compositionality)

Composite semantics is independent of component syntax:

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

Meta-theory in one line

Lemma (substitution)

Syntactic substitution corresponds to semantic composition:

substitution monoid homomorphism

$$[\![M\,[\theta]]\!] = [\![-[-]\,[M, \theta]]\!] \overset{\downarrow}{=} -[-]\left[[\![M]\!], [\![\theta]\!] \right] \coloneqq [\![M]\!] \circ [\![\theta]\!]$$

Lemma (compositionality)

Composite semantics is independent of component syntax:

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

Meta-theory in one line

Lemma (substitution)

Syntactic substitution corresponds to semantic composition:

$$[\![M\,[\theta]]\!] = [\![-[-]\,[M,\theta]]\!] \overset{\substack{\text{substitution monoid homomorphism}\\ \downarrow}}{=} -[-]\left[[\![M]\!]\,,[\![\theta]\!]\right] \coloneqq [\![M]\!]\circ[\![\theta]\!]$$

Lemma (compositionality)

Composite semantics is independent of component syntax:

$$\begin{aligned}[\![C[M]]\!] &= [\![(?m\,[M]) \gg\!= (m \mapsto C[-])]\!] \underset{\substack{\uparrow\\ \gg= \text{ is}\\ \text{homomorphic}\\ \text{extension}}}{=} [\![?m\,[M]]\!] \gg\!= (m \mapsto [\![C[-]]\!])\\ &=: \mathrm{plug}([\![C[-]]\!]\,,[\![M]\!])\end{aligned}$$

Talk structure

- Contribution
- Substitution monoids
- MAST in detail
- WIP

MAST summary: semantic domain for syntax and semantics

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

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

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

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

  P-element: $\quad p \in P_s\Gamma$

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

  identifying, e.g.:

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

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

MAST summary: semantic domain for syntax and semantics

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

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

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

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

  P-element: $p \in P_s\Gamma$

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

  identifying, e.g.:

$$[p[\mathsf{weaken}], \theta]_{\Delta_1 + \Delta_2} = [p, \theta \circ \rho]_{\Delta_1}$$
$$\left[p[x', x'' \mapsto x]_{x\in\Delta}, \theta\right]_\Delta = [p, \theta + \theta]_{\Delta + \Delta}$$

Scope-change as tensorial strength

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

MAST summary: semantic domain for syntax and semantics

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

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

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

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

  P-element: $p \in P_s\Gamma$

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

  identifying, e.g.:

  $$[p[\mathsf{weaken}], \theta]_{\Delta_1 + \Delta_2} = [p, \theta \circ \rho]_{\Delta_1}$$
  $$\left[p[x', x'' \mapsto x]_{x\in\Delta}, \theta\right]_\Delta = [p, \theta + \theta]_{\Delta + \Delta}$$

Scope-change as tensorial strength

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

Substitution monoids

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

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

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

MAST: semantic domain for syntax

Signature functors

$$\begin{tikzcd} \mathbf{R\text{-}Struct} \arrow[loop left, "\mathbf{O}"] \end{tikzcd}$$

Scope-change as tensorial strength

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

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

$$P \otimes_{\bullet} \left(\mathsf{var} \begin{array}{c}\mathbb{I}\\ \downarrow \\ A\end{array}\right) := P \otimes A$$

Example

Sequential fragment signature functor:

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

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

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

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

Takeaway (modularity)

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

Signatrue combinators

[cf. SOAS]

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

Example (Binding signatures [Actzel'78])

$\mathsf{Seq} \cong$

$$\coprod_{A \in \mathsf{Type}} \begin{pmatrix} (\mathbf{let}\ x : A = \_\ \mathbf{in}\ \_) : \\ \underset{\mathsf{comp}\ B}{\looparrowright} ((@\mathsf{comp}\ A) \times ([x : A] \triangleright)) \end{pmatrix}$$

NB

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

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

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

MAST: semantic domain for syntax

Abstract syntax: inductive representation

Every initial algebra:

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

Supports standard definitions:

$$\begin{tikzcd}
 & \mathbf{H} \arrow[d, "?{-}[{-}]"] & \\
\$\mathbf{H} \otimes \$\mathbf{H} \arrow[r, "{-}[{-}]"] & \$\mathbf{H} & \mathbb{I} \arrow[l, "\mathsf{var}"'] \\
 & \mathbf{O}(\$\mathbf{H}) \arrow[u, "⟦{-}⟧"] &
\end{tikzcd}$$

Independently of concrete representation, e.g.,:

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

MAST: semantic domain for semantics

Example

$\mathbf{M} = \left(\mathcal{C}, \mathrm{T}, \mathrm{return}, \gg\!=, [\![-]\!]\right)$:

- $\mathcal{C}$: Cartesian category with chosen finite products
- $\left(\mathrm{T}, \mathrm{return}, \gg\!=\right)$ strong monad over $\mathcal{C}$
- $[\![-]\!]$: Type $\to \mathcal{C}$ type interpretation

induces:

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

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

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

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:

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

- Variables: $\mathsf{var} : \big((x : A) \in \Gamma \mapsto \big(\llbracket\Gamma\rrbracket \xrightarrow{\pi_x} \llbracket A\rrbracket\big)\big)$

  (1^{st}-class sorts only)

MAST: compatibility

Substitution-compatible algebra

$[\![-]\!] : \mathbf{OM} \to \mathbf{M}$:

$$\begin{tikzcd}
 & \underline{\mathbf{O}}(\underline{\mathbf{M}} \otimes \underline{\mathbf{M}}) \arrow[dr, "\underline{\mathbf{O}}(-[-]_{\mathbf{M}})"] & \\
(\underline{\mathbf{OM}}) \otimes_\bullet \mathsf{var}^{\mathbf{M}} \arrow[ur, "\mathsf{str}"] \arrow[dr, "{[\![-]\!] \otimes_\bullet \mathsf{id}}"'] \arrow[rr, phantom, "\text{compatibility} \atop ="] & & \underline{\mathbf{OM}} \arrow[d, "{[\![-]\!]}"] \\
 & \underline{\mathbf{M}} \otimes_\bullet \mathsf{var}^{\mathbf{M}} \arrow[r, "-[-]_{\mathbf{M}}"'] & \underline{\mathbf{M}}
\end{tikzcd}$$

Example (Seq-algebra)

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

$$[\![\Gamma]\!] \xrightarrow{(\mathsf{id}, f)} [\![\Gamma]\!] \times \mathrm{T}\,[\![A]\!] \xrightarrow{\gg\!= g} \mathrm{T}\,[\![B]\!]$$

Takeaway

Equip each semantic interpretation with its compatibility proof

Example (Seq-compatibility)

Compatibility:

$$\begin{tikzcd}
{[\![\Gamma]\!]} \arrow[r, "{(\mathsf{id}, (f \circ \theta))}"] \arrow[d, "\theta"'] \arrow[dr, phantom, "\text{products} \atop ="] & {[\![\Gamma]\!] \times \mathrm{T}\,[\![A]\!]} \arrow[d, "\theta \times \mathsf{id}"] \arrow[dr, bend left, "\gg\!= (g \circ (\theta \times \mathsf{id}))"] \arrow[dr, phantom, "\text{strong monad laws} \atop ="] & \\
{[\![\Delta]\!]} \arrow[r, "{(\mathsf{id}, f)}"'] & {[\![\Delta]\!] \times \mathrm{T}\,[\![A]\!]} \arrow[r, bend right, "\gg\!= g"'] & \mathrm{T}\,[\![B]\!]
\end{tikzcd}$$

Substitution $\mathbf{O}$-monoid

Substitution monoid with compatible $\mathbf{O}$-algebra structure

$$
\begin{tikzcd}
 & & \bullet & & \\
\bullet \arrow[urr, "\subseteq"] & & \bullet \arrow[u, "\subseteq"] & & \bullet \arrow[ull, "\subseteq"'] \\
\cup\{\mathsf{Seq}\} \arrow[u, "\subseteq"] \arrow[urr, "\subseteq"] & & \cup\{\mathsf{Fun}\} \arrow[ull, "\subseteq"'] \arrow[urr, "\subseteq"] & & \cup\{\mathsf{Record}\} \arrow[u, "\subseteq"] \arrow[ull, "\subseteq"'] \\
 & & \{\mathsf{Base}\} \arrow[ull, "\subseteq"'] \arrow[u, "\subseteq"] \arrow[urr, "\subseteq"] & &
\end{tikzcd}
$$

Want more?

In the paper:

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

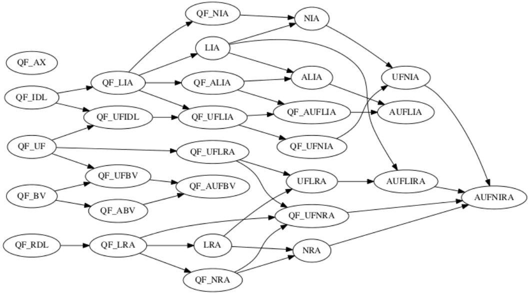
Talk structure

- 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]

Contribution

Modular Abstract Syntax Trees (MAST)

- SOAS $\overset{\text{generalise}}{\rightsquigarrow}$ 2^{nd}-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]