Modular abstract syntax trees (MAST)

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

Paper:

Slides:

ACP Seminar, SDU Odense (28 October 2026)

Funded by: THE ROYAL SOCIETY, ARIA, eTSS, erc, European Research Council Established by the European Commission, FWF Austrian Science Fund

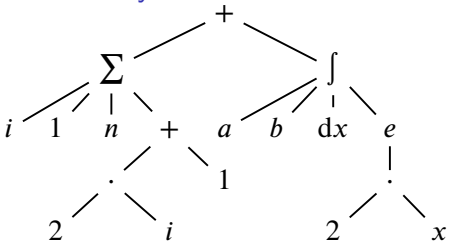
Syntax representation

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

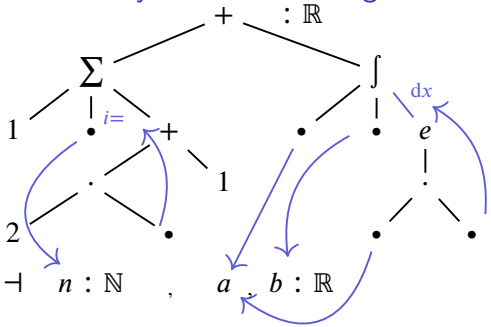
Concrete syntax

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

Abstract syntax



Abstract syntax with binding



Contribution: MAST

generic mathematical & computational
representation
of
intrinsically sorted, well-scoped
abstract syntax
with
binding, substitution, and holes
& their
semantic meta-theory

Approach

Second-Order Abstract Syntax (SOAS) [Fiore, Plotkin, and Turi '99] with
1^{st}-class + 2^{nd}-class sort-distinction
cf. Uemura's ['19] representable / non-representable sort-distinction

Contribution: MAST

generic mathematical & computational
representation
of
intrinsically sorted, well-scoped
abstract syntax
with
binding, substitution, and holes
& their
semantic meta-theory

$\Gamma = [x_1 : A_1, \ldots, x_n : A_n]$

$\mathsf{val}\, B\, \Gamma \qquad \mathsf{comp}\, B\, \Gamma$

$A_1, \ldots, A_n$ 1^{st}-class sorts only
B any sort

Approach

Second-Order Abstract Syntax (SOAS) [Fiore, Plotkin, and Turi '99] with
1^{st}-class + 2^{nd}-class sort-distinction
cf. Uemura's ['19] representable / non-representable sort-distinction

Contribution: MAST

generic mathematical & computational
representation
of
intrinsically sorted, well-scoped
abstract syntax
with
binding, substitution, and holes
& their
semantic meta-theory

object-level equations

$$m : (x : A \vdash \mathsf{comp}\, B), v : A \vdash (\lambda x.?\, m\,[x])(\mathsf{val}?v) = ?\, m\,[x \mapsto ?v]$$

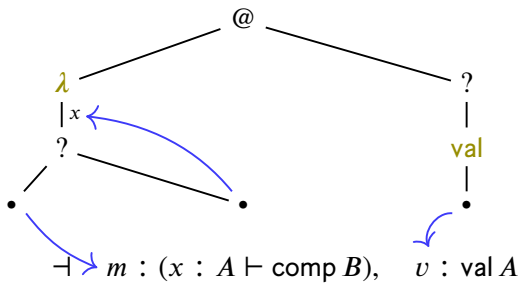
Approach

Second-Order Abstract Syntax (SOAS) [Fiore, Plotkin, and Turi '99] with
1^{st}-class + 2^{nd}-class sort-distinction
cf. Uemura's ['19] representable / non-representable sort-distinction

Contribution: MAST

generic mathematical & computational
representation
of
intrinsically sorted, well-scoped
abstract syntax
with
binding, substitution, and holes
& their
semantic meta-theory

e.g., $(\lambda x.?\, m\,[x])(\mathsf{val}?v)$:



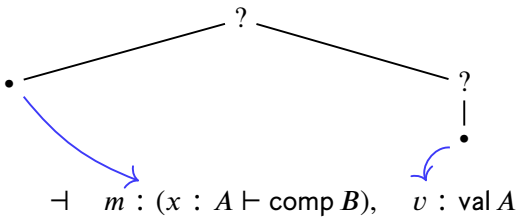
Approach

Second-Order Abstract Syntax (SOAS) [Fiore, Plotkin, and Turi '99] with
1^{st}-class + 2^{nd}-class sort-distinction
cf. Uemura's ['19] representable / non-representable sort-distinction

Contribution: MAST

generic mathematical & computational
representation
of
intrinsically sorted, well-scoped
abstract syntax
with
binding, substitution, and holes
& their
semantic meta-theory

e.g., $? \, m \, [x \mapsto ? v]$:



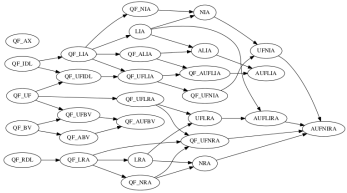
Approach

Second-Order Abstract Syntax (SOAS) [Fiore, Plotkin, and Turi '99] with
1^{st}-class + 2^{nd}-class sort-distinction
cf. Uemura's ['19] representable / non-representable sort-distinction

Contribution: MAST

generic mathematical & **computational**
representation
of
intrinsically sorted, well-scoped
abstract syntax
with
binding, substitution, and holes
& their
semantic meta-theory

Expression Problem (wip)
Idris SMTLIB FFI with modular
serialisation, deserialisation,
type-reconstruction
via generic traversals & parser combinators



Approach

Second-Order Abstract Syntax (SOAS) [Fiore, Plotkin, and Turi '99] with
1^{st}-class + 2^{nd}-class sort-distinction
cf. Uemura's ['19] representable / non-representable sort-distinction

Contribution: MAST

generic **mathematical** & computational
representation
of
intrinsically sorted, well-scoped
abstract syntax
with
binding, substitution, and holes
& their
semantic meta-theory

semantics of abstract syntax
via initiality of abstract syntax
substitution derived via traversal

CBV case study:
128 substitution & compositionality
lemmata

Approach

Second-Order Abstract Syntax (SOAS) [Fiore, Plotkin, and Turi '99] with
1^{st}-class + 2^{nd}-class sort-distinction
cf. Uemura's ['19] representable / non-representable sort-distinction

Contribution: MAST

generic mathematical & computational
representation
of
intrinsically sorted, well-scoped
abstract syntax
with
binding, substitution, and holes
& their
semantic meta-theory

one generic representation & proofs in
library code

instead of

dynamically generate code via
metaprogramming / tactics

Approach

Second-Order Abstract Syntax (SOAS) [Fiore, Plotkin, and Turi '99] with
1^{st}-class + 2^{nd}-class sort-distinction
cf. Uemura's ['19] representable / non-representable sort-distinction

Contribution: MAST

generic mathematical & computational
representation
of
intrinsically sorted, well-scoped
abstract syntax
with
binding, substitution, and holes
& their
semantic meta-theory



MAST tutorial

Future work:

- Realisability [Brown's thesis]
- Type variables [Fiore & Hamana'13]
- Measurability

Approach

Second-Order Abstract Syntax (SOAS) [Fiore, Plotkin, and Turi '99] with
1^{st}-class + 2^{nd}-class sort-distinction

Talk structure

- Contribution
- Applications: SMTLIB FFI and CBV á la carte
- Substitution actions/modules
- MAST in detail

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

But:

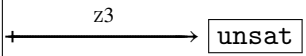
- Without recompiling previous modules; alternatively
- Retaining and reusing both old and new languages

Some solutions

- Scala Mix-ins [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]

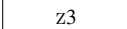
Standardised query language for SMT solvers such as Z3 [de Moura-Bjørner'08]:

```
; Integer arithmetic
(set-logic QF_LIA)
(declare-const x Int)
(declare-const y Int)
(assert (= (- x y)
           (+ x (- y) 1)))
(check-sat)
; unsat
(exit)
```



Standardised query language for SMT solvers such as Z3 [de Moura-Bjørner'08]:

```
; Integer arithmetic
(set-logic QF_LIA)
(declare-const x Int)
(declare-const y Int)
(assert (= (- x y)
           (+ x 1 (- y) (- 1))))
(check-sat)
(get-value (x y))
(exit)
```




```
sat
((x 0)
 (y 0))
```

Satyr: Idris FFI for SMT-LIB (WiP)

Want Z3 queries FFI, e.g.:

- Python's z3py
- Haskell's SBV
- Agda's Schmitty
- ...

Work in progress:



Intrinsically-typed well-scoped FFI with holes and modular serialisation and deserialisation

Intrinsically-typed well-scoped syntax with splicing[1]

```
query : Term BoolSMT [< "x" :- IntSMT, "y" :- IntSMT]
query = (V "x" - V "y") .=. (V "x" + 1 + (- V "y") + (- 1))
```

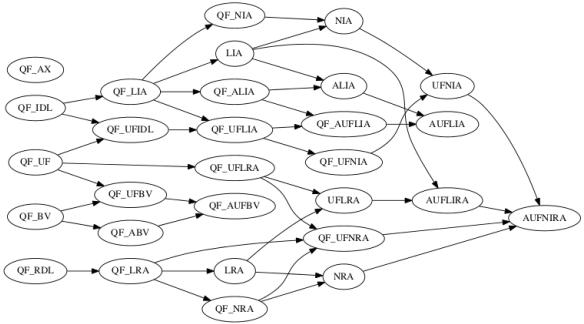
- No dangling variables
- Only represents well-typed queries
- Hole support (not shown)

[1] Illustration purposes only

Need for Modularity

SMT-LIB query language

- S-expressions
- 29 theories
- multiple syntax extensions



Data-types á la carte

[Swierstra'08]

Initial algebra semantics

[Goguen and Thatcher'74]

Abstract syntax (without binding) in Haskell

(Idris implementation on next slide)

Terms á la carte in Idris

```
Signature : Type
Signature = Type -> Type

BinOp : Signature
BinOp x = (x,x)

(+) : (x,y : Signature) -> Signature
(s + t) x = Either (s x) (t x)

IntSig : Signature
IntSig = BinOp + BinOp + const Integer
```

```
Num (IntSig .Term x) where
  x + y = Op (Left (Left  (x,y)))
  x * y = Op (Left (Right (x,y)))
  fromInteger x = Op (Right x)

ex1 : IntSig .Term String
ex1 = V "x" + V "y"
```

```
data (.Term) : Signature -> Type -> Type where
  V  : a -> sig.Term a
  Op : sig (sig.Term a) -> sig.Term a
```

Terms á la carte in Idris

```
Signature : Type
Signature = Type -> Type

BinOp : Signature
BinOp x = (x,x)

(+) : (x,y : Signature) -> Signature
(s + t) x = Either (s x) (t x)

IntSig : Signature
IntSig = BinOp + BinOp + const Integer
```

```
Num (IntSig .Term x) where
  x + y = Op (Left (Left  (x,y)))
  x * y = Op (Left (Right (x,y)))
  fromInteger x = Op (Right x)

ex1 : IntSig .Term String
ex1 = V "x" + V "y"
```

```
data (.Term) : Signature -> Type -> Type where
  V  : a -> sig.Term a
  Op : sig (sig.Term a) -> sig.Term a
```

Evaluator á la carte in Idris

```
(.AlgebraOver) : Signature -> Type -> Type
sig.AlgebraOver a = sig a -> a

plus : BinOp .AlgebraOver Integer
plus (x,y) = x + y

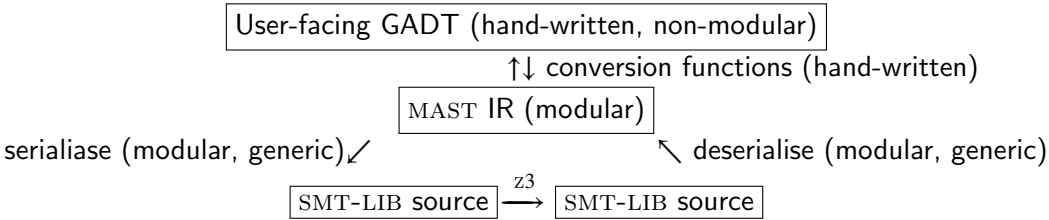
times : BinOp .AlgebraOver Integer
times (x,y) = x * y

constant : (const a).AlgebraOver a
constant x = x

(:+:) : s.AlgebraOver a -> t.AlgebraOver a -> (s+t).AlgebraOver a
(salg :+: talg) (Left  sVal) = salg sVal
(salg :+: talg) (Right tVal) = talg tVal

intAlg : IntSig .AlgebraOver Integer
intAlg = ((plus :+: times)    {s = BinOp, t = BinOp}
                :+: constant) {s = BinOp + BinOp, t = const Integer}
```

Satyr Architecture



User-facing interface[2]

```
data Term : TypeSMT -> Context -> Type where
  (.=.) : Term a ctx -> Term a ctx
       -> Term BoolSMT ctx

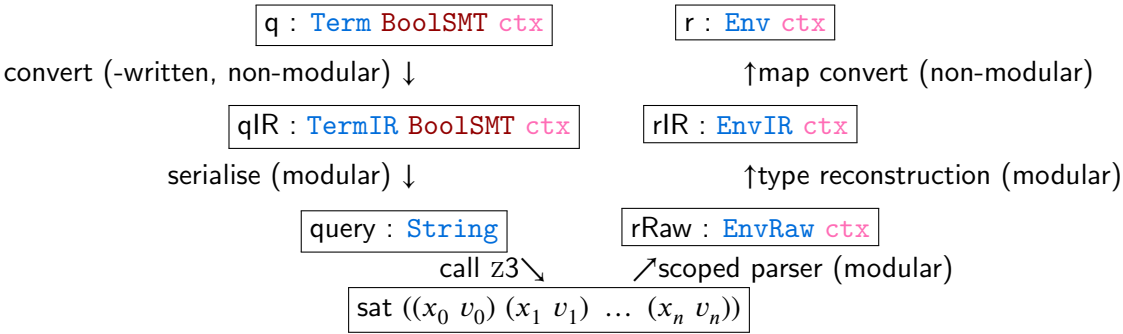
  (-),(+),(*) : Term IntSMT ctx -> Term IntSMT ctx
              -> Term IntSMT ctx

  (.-) :   Term IntSMT ctx
        -> Term IntSMT ctx

  Var  : (pos : (s :- ty) `Elem` ctx)
      -> Term ty ctx
  Const : Integer -> Term IntSMT ctx
```

[2]Illustration purposes only.

Architecture: query[3]



[3]Illustration purposes only.

MAST traversals[4]

```
fold alg ren subst var metavarEnv : Term tySMT ctx → b tySMT ctx
```

- `b`: SMT-LIB-type-indexed, context-indexed family
- `alg`: algebra, i.e., per-construct function
- `ren`: how to rename variables in the algebra
- `subst`: how to substitute in the algebra
- `metavarEnv`: what to splice into holes (if any)

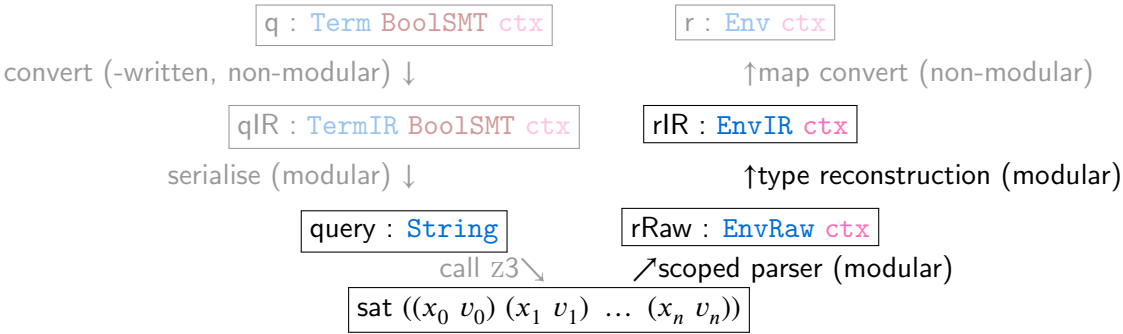
[4] Illustration purposes only

Serialisation via MAST traversal

- Serialise with user-preferred names.
- SMT-LIB is nominal, MAST is de Bruijn. Serialisation avoids capture if needed via mangling.

  Open problem(?): nominal syntax without shadowing avoiding per-name de Bruijn?
- Renaming permutes the preferred-name dictionary
- Substitution serialises to object-level `let`-binding.

Architecture: query[3]



[3]Illustration purposes only.

TAP: linear-time parser-combinators [Krishnaswami and Yallop'19]

Typed algebraic approach to parsing

Linear time parser combinator library through:

- μ-regexes: regexes + guarded fixpoints
- restricted by static analysis that ensures:
  - grammatical disambiguity
  - 1-lookahead recursive descent parsability

Dependently-typed extension [Greg Brown]

- Extend parser with dependently-typed state.
- Parsers can emit well-scoped raw terms.
- Per-construct parser fragments
- Combined to per-theory parsers.

Type reconstruction

Simple constraint solving

- Relatively straightforward.
- Reconstruct types per-construct.
- Per-construct solver fragments.
- Combined to per-theory solver.

Fully generic inference requires **type variable** support

Summary



Intrinsically-typed well-scoped FFI with holes and modular serialisation and deserialisation

`q : Term BoolSMT ctx` — `r : Env ctx`

convert (-written, non-modular) ↓ — ↑map convert (non-modular)

`qIR : TermIR BoolSMT ctx` — `rIR : EnvIR ctx`

serialise (modular) ↓ — ↑type reconstruction (modular)

`query : String` — `rRaw : EnvRaw ctx`

call Z3↘ — ↗scoped parser (modular)

sat $((x_0\ v_0)\ (x_1\ v_1)\ \ldots\ (x_n\ v_n))$

Talk structure

- Contribution
- Applications: SMTLIB FFI and CBV á la carte
- Substitution actions/modules
- MAST in detail

Call-by-Value λ-calculus

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

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

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

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

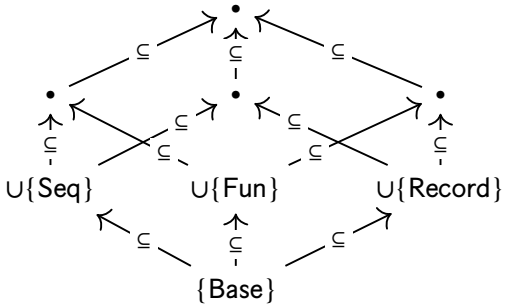
fragment	syntactic constructs	types	semantics
base	returning a value: val		strong monad over a Cartesian category
sequential	sequencing: **let**		
functions	abst., app. $(\lambda x. : A), (@)$	function $(\to)$	Kleisli exponentials
variants	c'tors, pattern match $A.C_i -$, **case** $-$ **of** $\{C_i x_i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

⋮

Dream

Iterative semantic development

- Add syntax
- Add semantics

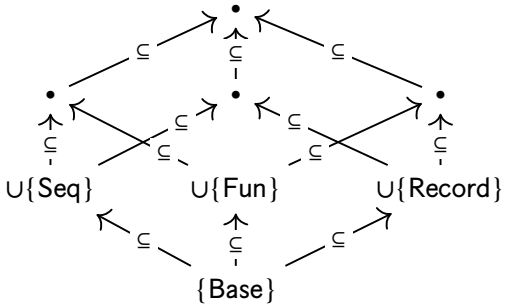


- Profit!

Dream vs. Reality

Iterative semantic development

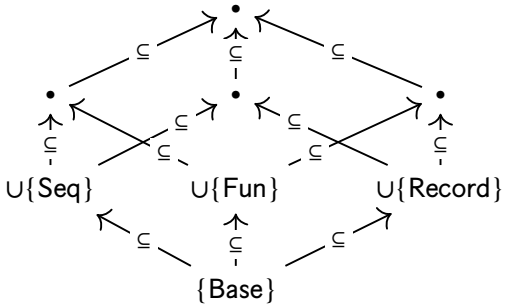
- Add syntax
- Add semantics
- Develop meta-theory:
  - Substitution lemma
  - Compositionality
  - Soundness
  - Adequacy
- Profit!



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 $M\,[\theta]\,[\sigma] = M\,[\theta \circ \sigma]$.
Typically several further inductions over all constructs for the semantic lemma. □

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. □

Dream

It would be nice if tedious bits were…

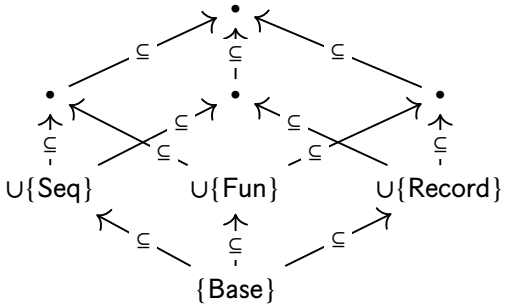
… free

Dream vs. Reality

It would be nice if tedious bits were…

… ~~free~~

… syntactically scaleable: additive syntactic work per new feature



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 system $\mathbf{R}$

Coproduct diagram in set:

$$\mathsf{Fst} \xrightarrow{\mathsf{fst}} \mathsf{Sort} \xleftarrow{\mathsf{snd}} \mathsf{Snd}$$

Example (CBV sorting system)

$$\mathrm{CBV}\mathsf{Type} \xrightarrow{\lambda x.x} \left(\mathrm{CBV}\mathsf{Type} \cup \{\mathsf{comp}\} \times \mathrm{CBV}\mathsf{Type}\right) \xleftarrow{\mathsf{comp}} \mathrm{CBV}\mathsf{Type}$$

Example (CBPV sorting system)

$$\mathrm{CBPV}\mathsf{ValueType} \xrightarrow{\lambda x.x} \mathrm{CBPV}\mathsf{Type} \xleftarrow{\lambda x.x} \mathrm{CBPV}\mathsf{CompType}$$

Technical contribution: generalise to heterogeneous sorts

	SOAS	MAST
Sorting system:	homogeneous	heterogeneous
	$\mathsf{Sort} \xrightarrow{\lambda x.x} \mathsf{Sort} \leftarrow \emptyset$	$\mathsf{Fst} \xrightarrow{\mathsf{fst}} \mathsf{Sort} \xleftarrow{\mathsf{snd}} \mathsf{Snd}$

$S_{\vdash}$: renaming cat. $\Gamma = [x_1 : s_1, \ldots, x_n : s_n] \xrightarrow{\Gamma \vdash \rho : \Delta} [y_1 : t_k, \ldots, y_k : t_k] = \Delta$

$P \in \mathbf{R\text{-}Strct} \coloneqq \mathbf{PSh}\left(\mathsf{Sort} \times \left(\mathsf{Sort}\right)_{\vdash}\right)$ $\quad \mathbf{PSh}\left(\mathsf{Sort} \times \left(\mathsf{Fst}\right)_{\vdash}\right) \cong \mathsf{Fst}\text{-}\mathbf{Strct} \times \mathbf{PSh}\left(\mathsf{Snd} \times \left(\mathsf{Fst}\right)_{\vdash}\right)$

$P : \mathsf{Sort} \times \mathsf{Sort}_{\vdash}^{\mathsf{op}} \to \mathbf{Set}$

$P : \mathsf{Sort} \times \mathsf{Fst}_{\vdash}^{\mathsf{op}} \to \mathbf{Set}$

$P|_{\mathsf{fst}} : \mathsf{Fst} \times \mathsf{Fst}_{\vdash}^{\mathsf{op}} \to \mathbf{Set}$

$P|_{\mathsf{snd}} : \mathsf{Snd} \times \mathsf{Fst}_{\vdash}^{\mathsf{op}} \to \mathbf{Set}$

Abstract syntax and denotational semantics: $\mathbb{S}, \mathbf{M} \in \mathbf{R\text{-}Strct}$

$\mathbb{S}_s\Gamma \coloneqq \{M \mid \Gamma \vdash M : s\}$ $\quad \mathbb{S}_s\left(\Gamma \xleftarrow{\rho} \Delta\right) \coloneqq \lambda(\Gamma \vdash M : s).(\Delta \vdash M[\rho] : s)$

$\mathbf{M}_s\Gamma \coloneqq \mathbf{C}(\llbracket\Gamma\rrbracket, \llbracket s\rrbracket)$ $\quad \mathbf{M}_s\left(\Gamma \xleftarrow{\rho} \Delta\right) \coloneqq \lambda\left(\llbracket\Gamma\rrbracket \xrightarrow{f} \llbracket s\rrbracket\right).\left(\llbracket\Delta\rrbracket \xrightarrow{\llbracket\rho\rrbracket} \llbracket\Gamma\rrbracket \xrightarrow{f} \llbracket s\rrbracket\right)$

Technical contribution: generalise to heterogeneous sorts

	SOAS	MAST
Sorting system:	homogeneous	heterogeneous
	$\mathsf{Sort} \xrightarrow{\lambda x.x} \mathsf{Sort} \leftarrow \emptyset$	$\mathsf{Fst} \xrightarrow{\mathsf{fst}} \mathsf{Sort} \xleftarrow{\mathsf{snd}} \mathsf{Snd}$
$P \in \mathbf{R\text{-}Strct} \coloneqq$	$\mathbf{PSh}\left(\mathsf{Sort} \times \left(\mathsf{Sort}\right)_{\vdash}\right)$	$\mathbf{PSh}\left(\mathsf{Sort} \times \left(\mathsf{Fst}\right)_{\vdash}\right) \cong \mathsf{Fst}\text{-}\mathbf{Strct} \times \mathbf{PSh}\left(\mathsf{Snd} \times \left(\mathsf{Fst}\right)_{\vdash}\right)$

Abstract syntax and denotational semantics: $\mathbb{S}, \mathbf{M} \in \mathbf{R\text{-}Strct}$

Variables $\mathbb{V}_s\Gamma \coloneqq \{x | (x : s) \in \Gamma\}$: $\mathbb{V} \in \mathbf{R\text{-}Strct}$ $\quad$ $\mathbb{V} \in \mathsf{Fst}\text{-}\mathbf{Strct}$

Technical contribution: generalise to heterogeneous sorts

	SOAS	MAST
Sorting system:	homogeneous	heterogeneous
	$\mathsf{Sort} \xrightarrow{\lambda x.x} \mathsf{Sort} \leftarrow \emptyset$	$\mathsf{Fst} \xrightarrow{\mathsf{fst}} \mathsf{Sort} \xleftarrow{\mathsf{snd}} \mathsf{Snd}$
$P \in \mathbf{R\text{-}Strct} \coloneqq$	$\mathbf{PSh}\left(\mathsf{Sort} \times \left(\mathsf{Sort}\right)_{\vdash}\right)$	$\mathbf{PSh}\left(\mathsf{Sort} \times \left(\mathsf{Fst}\right)_{\vdash}\right) \cong$ $\mathsf{Fst}\text{-}\mathbf{Strct} \times \mathbf{PSh}\left(\mathsf{Snd} \times \left(\mathsf{Fst}\right)_{\vdash}\right)$

Abstract syntax and denotational semantics: $\mathbb{S}, \mathbf{M} \in \mathbf{R\text{-}Strct}$

Variables $\mathbb{V}_s\Gamma \coloneqq \{x | (x : s) \in \Gamma\}$: $\mathbb{V} \in \mathbf{R\text{-}Strct}$ $\quad$ $\mathbb{V} \in \mathsf{Fst}\text{-}\mathbf{Strct}$

Signature functors $\quad$ $\mathbf{o}\circlearrowright \mathbf{R\text{-}Strct}$ $\quad$ family of constructors + renaming action + functoriality

Technical contribution: generalise to heterogeneous sorts

	SOAS	MAST
Sorting system:	homogeneous	heterogeneous
	$\mathsf{Sort} \xrightarrow{\lambda x.x} \mathsf{Sort} \leftarrow \emptyset$	$\mathsf{Fst} \xrightarrow{\mathsf{fst}} \mathsf{Sort} \xleftarrow{\mathsf{snd}} \mathsf{Snd}$
$P \in \mathbf{R\text{-}Strct} \coloneqq$	$\mathbf{PSh}\left(\mathsf{Sort} \times \left(\mathsf{Sort}\right)_{\vdash}\right)$	$\mathbf{PSh}\left(\mathsf{Sort} \times \left(\mathsf{Fst}\right)_{\vdash}\right) \cong$ $\mathsf{Fst}\text{-}\mathbf{Strct} \times \mathbf{PSh}\left(\mathsf{Snd} \times \left(\mathsf{Fst}\right)_{\vdash}\right)$

Abstract syntax and denotational semantics: $\mathbb{S}, \mathbf{M} \in \mathbf{R\text{-}Strct}$

Variables $\mathbb{V}_s\Gamma \coloneqq \{x | (x : s) \in \Gamma\}$: $\mathbb{V} \in \mathbf{R\text{-}Strct}$ $\quad$ $\mathbb{V} \in \mathsf{Fst}\text{-}\mathbf{Strct}$

Signature functors $\quad \mathbf{O} \circlearrowright \mathbf{R\text{-}Strct}$ $\quad$ family of constructors + renaming action + functoriality

Pointed strength: $\quad P, A \in \mathbf{R\text{-}Strct}, \mathsf{var} : \mathbb{V} \to A$ $\quad$ $P \in \mathbf{R\text{-}Strct}, A \in \mathsf{Fst}\text{-}\mathbf{Strct}$

$\mathsf{str}_{P,\mathsf{var}} : (\mathbf{O}P) \otimes A \to \mathbf{O}(P \otimes A)$ $\quad$ $\mathsf{var} : \mathbb{V} \to A$

Technical contribution: generalise to heterogeneous sorts

	SOAS	MAST
Sorting system:	homogeneous	heterogeneous
	$\mathsf{Sort} \xrightarrow{\lambda x.x} \mathsf{Sort} \leftarrow \emptyset$	$\mathsf{Fst} \xrightarrow{\mathsf{fst}} \mathsf{Sort} \xleftarrow{\mathsf{snd}} \mathsf{Snd}$
$P \in \mathbf{R\text{-}Strct} \coloneqq$	$\mathbf{PSh}\left(\mathsf{Sort} \times \left(\mathsf{Sort}\right)_{\vdash}\right)$	$\mathbf{PSh}\left(\mathsf{Sort} \times \left(\mathsf{Fst}\right)_{\vdash}\right) \cong$ $\mathsf{Fst}\text{-}\mathbf{Strct} \times \mathbf{PSh}\left(\mathsf{Snd} \times \left(\mathsf{Fst}\right)_{\vdash}\right)$

Abstract syntax and denotational semantics: $\mathbb{S}, \mathbf{M} \in \mathbf{R\text{-}Strct}$

Variables $\mathbb{V}_s\Gamma \coloneqq \{x | (x : s) \in \Gamma\}$: $\mathbb{V} \in \mathbf{R\text{-}Strct}$ $\quad$ $\mathbb{V} \in \mathsf{Fst}\text{-}\mathbf{Strct}$

Signature functors: $\mathbf{O} \circlearrowright \mathbf{R\text{-}Strct}$ $\quad$ family of constructors + renaming action + functoriality

Pointed strength: $P, A \in \mathbf{R\text{-}Strct}, \mathsf{var} : \mathbb{V} \to A$ $\quad$ $P \in \mathbf{R\text{-}Strct}, A \in \mathsf{Fst}\text{-}\mathbf{Strct}$, $\mathsf{var} : \mathbb{V} \to A$

$\mathsf{str}_{P,\mathsf{var}} : (\mathbf{O}P) \otimes A \to \mathbf{O}(P \otimes A)$

Substitution: monoid $\quad$ action/module

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

$$\mathbf{M}|_{\mathsf{fst}} \otimes \mathbf{M}|_{\mathsf{fst}} \xrightarrow{-[-]} \mathbf{M}|_{\mathsf{fst}} \xleftarrow{\mathsf{var}} \mathbb{V}$$

$$\mathbf{M}|_{\mathsf{snd}} \otimes \mathbf{M}|_{\mathsf{fst}} \xrightarrow{-[-]} \mathbf{M}|_{\mathsf{snd}}$$

Technical contribution in MAST paper

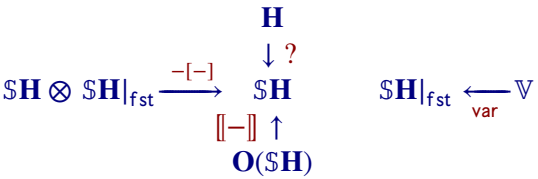
Modular Abstract Syntax Trees (MAST)

- SOAS $\overset{\text{generalise}}{\rightsquigarrow}$ 2^{nd}-class sorts
  - Kleisli bicategories [Gambino, Fiore, Hyland, and Winskel'19]
  - Generalise monoidal categories to actegories
  - Generalise substitution monoids to actions / modules
- MAST tutorial
- Case-study: CBV semantics á la carte (128 substitution lemmata)

Capstone: abstract syntax and substitution universality

Theorem (representation)

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



Plugging holes/metavariable substitution

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

Capstone: semantics

Key propaganda

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

$$\begin{array}{ccc} \mathbf{M} \otimes \mathbf{M}|_{\mathsf{fst}} & \xrightarrow{-[-]} & \mathbf{M} \\ & & [\![-]\!] \uparrow \\ & & \mathbf{OM} \end{array} \qquad \mathbf{M}|_{\mathsf{fst}} \xleftarrow[\mathsf{var}]{} \mathbb{V}$$

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

$$\left(\mathbb{S}\mathbf{H}, -[-], \mathsf{var}, [\![-]\!], ?_{\mathbb{S}\mathbf{H}}\right) \xrightarrow{[\![-]\!]} \left(\mathbf{M}, -[-], \mathsf{var}, [\![-]\!], ?_{\mathbf{M}}\right) \qquad \left(\mathbf{H} \xrightarrow{?_{\mathbf{M}}} \mathbf{M}\right)$$

Meta-theory in one line

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 in one line

Lemma (substitution)

Syntactic substitution corresponds to semantic composition:

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

Lemma (compositionality)

Composite semantics is independent of component syntax:

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

Meta-theory in one line

Lemma (substitution)

Syntactic substitution corresponds to semantic composition:

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

Lemma (compositionality)

Composite semantics is independent of component syntax:

$$[\![C[M]]\!] = [\![C[?m] \gg\!= (m \mapsto M)]\!] \overset{\gg\!= \text{ is homomorphic extension}}{=} [\![C[?m]]\!] \gg\!= (m \mapsto [\![M]\!]) =: \mathrm{plug}([\![C[-]]\!], [\![M]\!])$$

Talk structure

- Contribution
- Applications: SMTLIB FFI and CBV á la carte
- Substitution actions/modules
- MAST in detail

MAST summary: semantic domain for syntax and semantics

MAST provides $(\mathbf{R} = \mathsf{Fst} \xrightarrow{\mathsf{fst}} \mathsf{Sort} \xleftarrow{\mathsf{snd}} \mathsf{Snd})$

- Contexts $\qquad \mathsf{Fst}_{\vdash} \ni \Gamma ::= [x_1 : s_1, \ldots, x_n : s_n]$
- Renamings $\qquad \mathsf{Fst}_{\vdash}(\Gamma, \Delta) \ni \Gamma \vdash \rho : \Delta$
- $\mathbf{R}$-structures: $\qquad \mathbf{PSh}(\mathsf{Sort} \times \mathsf{Fst}_{\vdash}) \ni P : \mathsf{Sort} \times \mathsf{Fst}_{\vdash}^{\mathrm{op}} \to \mathbf{Set}$
  $P_s\Gamma \ni p$: sort s element with variables in Γ
- Variables structure: $\qquad \mathbf{R\text{-}Strct} \ni \mathbb{V}_s\Gamma := \{x | (x : s) \in \Gamma\}$
- substitution tensor and the pointed action:
  $(\otimes) : \mathbf{R\text{-}Strct} \times \mathsf{Fst}\text{-}\mathbf{Strct} \to \mathbf{R\text{-}Strct}$ $\qquad (\otimes_\bullet) : \mathbf{R\text{-}Strct} \times (\mathbb{V}/\mathsf{Fst}\text{-}\mathbf{Strct}) \to \mathbf{R\text{-}Strct}$
  $(P \otimes Q)_s\Gamma \ni [p, \theta]_\Delta$: $\qquad P \otimes_\bullet \left(\mathsf{var} \overset{\mathbb{V}}{\underset{A}{\downarrow}}\right) := P \otimes A$
  P-element: $p \in P_s\Delta$ identifying, e.g.: $[p[\mathsf{weaken}], \theta]_{\Delta_1 + \Delta_2} = [p, \theta \circ \rho]_{\Delta_1}$
  Q-closure : $\theta \in \prod_{(y:r)\in\Delta} Q_r\Gamma$ $\qquad \left[p[x', x'' \mapsto x]_{x\in\Delta}, \theta\right]_\Delta = [p, \theta + \theta]_{\Delta + \Delta}$

Scope-change as tensorial strength

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

MAST summary: semantic domain for syntax and semantics

MAST provides $(\mathbf{R} = \mathsf{Fst} \xrightarrow{\mathsf{fst}} \mathsf{Sort} \xleftarrow{\mathsf{snd}} \mathsf{Snd})$

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

  $P_s\Gamma \ni p$: sort s element with variables in Γ
- Variables structure: $\qquad \mathbf{R}\text{-}\mathbf{Strct} \ni \mathbb{V}_s\Gamma := \{x | (x : s) \in \Gamma\}$
- substitution tensor and the pointed action:

  $(\otimes) : \mathbf{R}\text{-}\mathbf{Strct} \times \mathsf{Fst}\text{-}\mathbf{Strct} \to \mathbf{R}\text{-}\mathbf{Strct}$ $\qquad (\otimes_\bullet) : \mathbf{R}\text{-}\mathbf{Strct} \times (\mathbb{V}/\mathsf{Fst}\text{-}\mathbf{Strct}) \to \mathbf{R}\text{-}\mathbf{Strct}$

  $(P \otimes Q)_s\Gamma \ni [p, \theta]_\Delta$: $\qquad P \otimes_\bullet \left(\mathsf{var} \overset{\mathbb{V}}{\downarrow} A\right) := P \otimes A$

  P-element: $p \in P_s\Delta$ $\qquad$ identifying, e.g.: $\quad [p[\mathsf{weaken}], \theta]_{\Delta_1 + \Delta_2} = [p, \theta \circ \rho]_{\Delta_1}$

  Q-closure : $\theta \in \prod_{(y:r)\in\Delta} Q_r\Gamma$ $\qquad \left[p[x', x'' \mapsto x]_{x\in\Delta}, \theta\right]_\Delta = [p, \theta + \theta]_{\Delta + \Delta}$

  I.e.:

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

MAST summary: semantic domain for syntax and semantics

MAST provides $(\mathbf{R} = \mathsf{Fst} \xrightarrow{\mathsf{fst}} \mathsf{Sort} \xleftarrow{\mathsf{snd}} \mathsf{Snd})$

- Contexts $\qquad \mathsf{Fst}_{\vdash} \ni \Gamma ::= [x_1 : s_1, \ldots, x_n : s_n]$
- Renamings $\qquad \mathsf{Fst}_{\vdash}(\Gamma, \Delta) \ni \Gamma \vdash \rho : \Delta$
- $\mathbf{R}$-structures: $\qquad \mathbf{PSh}(\mathsf{Sort} \times \mathsf{Fst}_{\vdash}) \ni P : \mathsf{Sort} \times \mathsf{Fst}_{\vdash}^{\mathrm{op}} \to \mathbf{Set}$
  $P_s\Gamma \ni p$: sort s element with variables in Γ
- Variables structure: $\qquad \mathbf{R\text{-}Strct} \ni \mathbb{V}_s\Gamma := \{x | (x : s) \in \Gamma\}$
- substitution tensor and the pointed action:
  $(\otimes) : \mathbf{R\text{-}Strct} \times \mathsf{Fst}\text{-}\mathbf{Strct} \to \mathbf{R\text{-}Strct}$ $\qquad (\otimes_\bullet) : \mathbf{R\text{-}Strct} \times (\mathbb{V}/\mathsf{Fst}\text{-}\mathbf{Strct}) \to \mathbf{R\text{-}Strct}$
  $(P \otimes Q)_s\Gamma \ni [p, \theta]_\Delta$: $\qquad P \otimes_\bullet \left(\mathsf{var} \overset{\mathbb{V}}{\underset{A}{\downarrow}}\right) := P \otimes A$
  P-element: $p \in P_s\Delta$ identifying, e.g.: $[p[\mathsf{weaken}], \theta]_{\Delta_1 + \Delta_2} = [p, \theta \circ \rho]_{\Delta_1}$
  Q-closure : $\theta \in \prod_{(y:r)\in\Delta} Q_r\Gamma$ $\qquad \left[p[x', x'' \mapsto x]_{x\in\Delta}, \theta\right]_\Delta = [p, \theta + \theta]_{\Delta + \Delta}$

Scope-change as tensorial strength

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

MAST summary: semantic domain for syntax and semantics

MAST provides $(\mathbf{R} = \mathsf{Fst} \xrightarrow{\mathsf{fst}} \mathsf{Sort} \xleftarrow{\mathsf{snd}} \mathsf{Snd})$

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

  $P_s\Gamma \ni p$: sort s element with variables in Γ
- Variables structure: $\qquad \mathbf{R\text{-}Strct} \ni \mathbb{V}_s\Gamma := \{x | (x : s) \in \Gamma\}$
- substitution tensor and the pointed action:

  $(\otimes) : \mathbf{R\text{-}Strct} \times \mathsf{Fst}\text{-}\mathbf{Strct} \to \mathbf{R\text{-}Strct} \qquad (\otimes_\bullet) : \mathbf{R\text{-}Strct} \times (\mathbb{V}/\mathsf{Fst}\text{-}\mathbf{Strct}) \to \mathbf{R\text{-}Strct}$

  $(P \otimes Q)_s\Gamma \ni [p, \theta]_\Delta$: $\qquad P \otimes_\bullet \left(\mathsf{var} \downarrow_A^{\mathbb{V}}\right) := P \otimes A$

  P-element: $p \in P_s\Delta$ identifying, e.g.: $[p[\mathsf{weaken}], \theta]_{\Delta_1 + \Delta_2} = [p, \theta \circ \rho]_{\Delta_1}$

  Q-closure : $\theta \in \prod_{(y:r)\in\Delta} Q_r\Gamma \qquad \left[p[x', x'' \mapsto x]_{x\in\Delta}, \theta\right]_\Delta = [p, \theta + \theta]_{\Delta + \Delta}$

Scope-change as tensorial strength

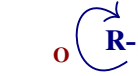
$\mathsf{str}^{\mathbf{O}} : (\mathbf{O}P) \otimes_\bullet \left(\mathsf{var} \downarrow_A^{\mathbb{V}}\right) \to \mathbf{O}\left(P \otimes_\bullet \left(\mathsf{var} \downarrow_A^{\mathbb{V}}\right)\right)$

Substitution actions

$\mathbf{M} \otimes \mathbf{M}|_{\mathsf{fst}} \xrightarrow{-[-]} \mathbf{M} \qquad \mathbf{M}|_{\mathsf{fst}} \xleftarrow{\mathsf{var}} \mathbb{V}$

MAST: semantic domain for syntax

Signature functors



Scope-change as tensorial strength

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

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

$$P\otimes_{\bullet}\left(\mathsf{var}\overset{\mathbb{V}}{\downarrow}_{A}\right) \coloneqq P \otimes A$$

Example

Sequential fragment signature functor:

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

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

Takeaway (modularity)

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

MAST: semantic domain for syntax

Signature combinators [cf. SOAS]

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

Example (Binding signatures [Actzel'78])

$\mathsf{Seq} \cong$

$$\coprod_{A,B \in \mathsf{Type}} \left(\begin{array}{l} (\mathbf{let}\ x : A = _\ \mathbf{in}\ _) : \underset{\mathsf{comp}\,B}{\nrightarrow} \\ \quad (@\mathsf{comp}\,A) \times \\ \qquad ([x : A] \triangleright - @\ \mathsf{comp}\,B) \end{array} \right)$$

NB

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

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

Abstract syntax: inductive representation

Every initial algebra: $\mathbb{S}^{\mathbf{O}}\mathbf{H} \coloneqq \boldsymbol{\mu} X.(\mathbf{O}X) \amalg \Rightarrow_{\mathsf{fst}} \mathbb{V} \amalg \mathbf{H} \otimes X|_{\mathsf{fst}}$

Supports standard definitions:

$$\mathbb{S}\mathbf{H} \otimes \mathbb{S}\mathbf{H}|_{\mathsf{fst}} \xrightarrow{-[-]} \mathbb{S}\mathbf{H} \qquad \mathbb{S}\mathbf{H}|_{\mathsf{fst}} \xleftarrow[\mathsf{var}]{} \mathbb{V}$$

$$\mathbf{H} \downarrow ? \qquad [\![-]\!] \uparrow \mathbf{O}(\mathbb{S}\mathbf{H})$$

Independently of concrete representation, e.g.,:

- De-Bruijn
- Nominal
- Locally nameless
- Co-de Bruijn (NB: inefficient substitution)
- Graphical

Example

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

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

induces:

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

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

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

Syntactic substitution monoid

$$\mathbb{S}^{\mathbf{O}}\mathbf{H} \otimes (\mathbb{S}^{\mathbf{O}}\mathbf{H})\big|_{\mathsf{fst}} \xrightarrow{-[-]} \mathbb{S}^{\mathbf{O}}\mathbf{H} \qquad \mathbb{S}^{\mathbf{O}}\mathbf{H}\big|_{\mathsf{fst}} \xleftarrow{\mathsf{var}} \mathbb{V}$$

Monoid axioms amount to syntactic substitution lemma

Example

Semantic substitution action:

$$\mathbf{M} \otimes \mathbf{M}|_{\mathsf{fst}} \xrightarrow{-[-]} \mathbf{M} \qquad \mathbf{M}|_{\mathsf{fst}} \xleftarrow{\mathsf{var}} \mathbb{V}$$

- Substitution via composition:

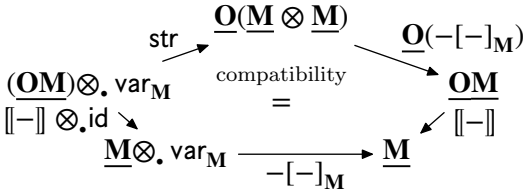
$$\left(\llbracket\Delta\rrbracket \xrightarrow{f} \llbracket s\rrbracket\right)\left[\llbracket\Gamma\rrbracket \xrightarrow{\theta} \llbracket\Delta\rrbracket\right] : \llbracket\Gamma\rrbracket \xrightarrow{\theta} \llbracket\Delta\rrbracket \xrightarrow{f} \llbracket s\rrbracket$$

- Variables: (1^{st}-class sorts only)

$$\mathsf{var} : \left((x : A) \in \Gamma \mapsto \left(\llbracket\Gamma\rrbracket \xrightarrow{\boldsymbol{\pi}_x} \llbracket A\rrbracket\right)\right)$$

Substitution-compatible algebra

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



Example (Seq-algebra)

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

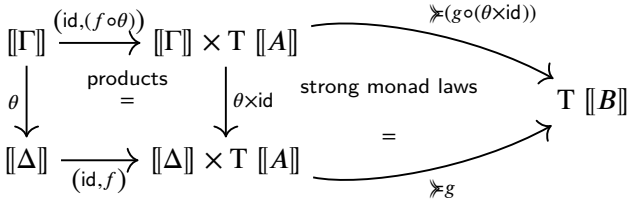
$$\llbracket\Gamma\rrbracket \xrightarrow{(\mathsf{id},f)} \llbracket\Gamma\rrbracket \times \mathrm{T}\ \llbracket A\rrbracket \xrightarrow{\gg g} \mathrm{T}\ \llbracket B\rrbracket$$

Takeaway

Equip each semantic interpretation with its compatibility proof

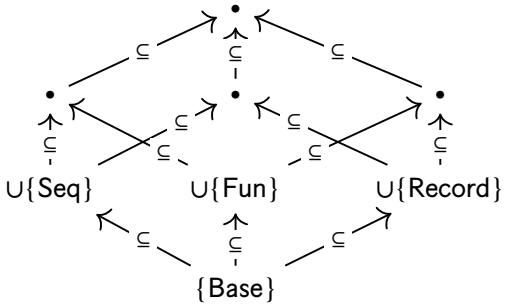
Example (Seq-compatibility)

Compatibility:



Substitution $\mathbf{O}$-action

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



Contribution: MAST

generic mathematical & computational
representation
of
intrinsically sorted, well-scoped
abstract syntax
with
binding, substitution, and holes
& their
semantic meta-theory



MAST tutorial

Future work:

- Realisability [Brown's thesis]
- Type variables [Fiore & Hamana'13]
- Measurability

Approach

Second-Order Abstract Syntax (SOAS) [Fiore, Plotkin, and Turi '99] with
1^{st}-class + 2^{nd}-class sort-distinction

NB: comparison with Uemura['19] (cf. [Graetzer & Sterling '22])

Disclaimer: apples and oranges

	Uemura	SOAS/MAST
goals	model theory of type theory	semantics of simply-typed syntax
method	logical framework (built-in $\prod$-types & Identity types+equality reflection& universe)	multi-sorted binding signatures
substitution	postulated for syntactic representation	derived from representation's induction principles
representable sorts	derived indirectly	specified directly