

# **High-level signatures and initial semantics**

Ambroise Lafont

joint work with Benedikt Ahrens, André Hirschowitz, Marco Maggesi

CSL 2018

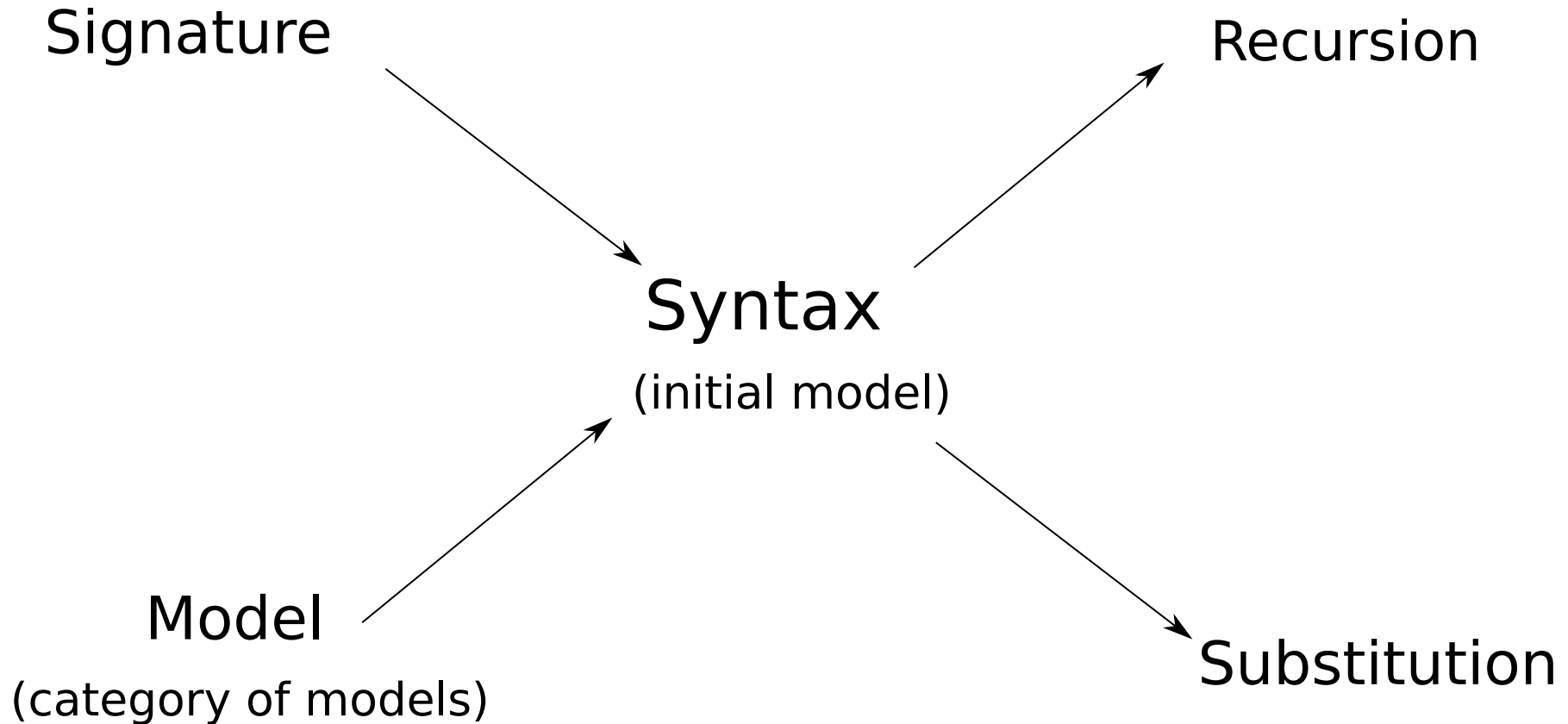
# Overview

**Purpose of our work:** to specify and construct untyped syntaxes with variables and a well-behaved substitution (e.g. lambda-calculus).

More specifically (terms in italics will be explained):

1. we work with a general notion of *signature* based on *monads* and *modules*. Not all of them do *generate a syntax*
2. classical *binding signatures* embed into our signatures as *algebraic signatures*, and indeed generate a syntax.
3. our main result: any *quotient* of algebraic signatures also generates a syntax

# What is a syntax?



**Signatures which we care about:** those whose category of models have an *initial object*, i.e., generate a syntax.

# Table of contents

- 1. Binding signatures and their models**
2. Signatures and models based on monads and modules
3. Presentables signatures

# Example: 0, ★

Consider the syntax generated by a binary operation ★ and a constant **0** (and variables):

$$\begin{array}{ll} \text{expr} ::= x & (\text{variable}) \\ \quad | t_1 \star t_2 & (\text{binary operation}) \\ \quad | 0 & (\text{constant}) \end{array}$$

The syntax can be considered as the endofunctor  $B$  (on Set):

$$B(X) = \text{expressions over } X$$

$$B(\emptyset) = \{0, 0 \star 0, \dots\}$$

$$B(\{x, y\}) = \{0, 0 \star 0, \dots, x, y, x \star y, \dots\}$$

# Example: 0, ★

The binary operation ★ induces a natural transformation:

$$B \times B \rightarrow B$$

The constant 0 induces a natural transformation:

$$1 \rightarrow B$$

Variables induce a natural transformation

$$\text{Id}_{\text{Set}} \rightarrow B$$

They gather into a single natural transformation:

$$B \times B + 1 + \text{Id}_{\text{Set}} \rightarrow B$$

i.e.  $B$  is an algebra for the endofunctor  $F \mapsto F \times F + 1 + \text{Id}_{\text{Set}}$  on the category  $\text{End}_{\text{Set}}$ .

Actually,  $B$  can be defined to be the initial algebra.

# Binding Signatures

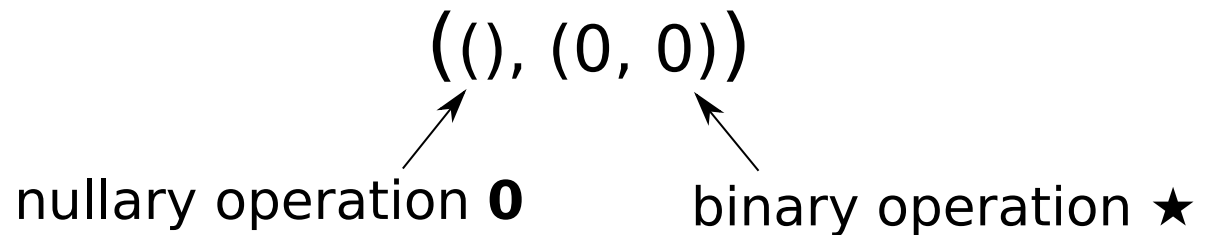
## Definition

**Binding signature** = a family of lists of natural numbers.

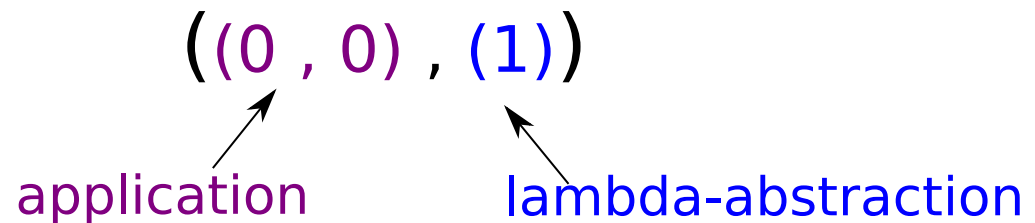
Each list specifies an operation in the syntax:

- length of the list = number of arguments of the operation
- natural number in the list = number of bound variables in the corresponding argument

**Syntax with 0, ★:**



**Lambda calculus:**



# Signatures and endofunctors

In the same spirit as in the first example  $(0, \star)$ , any binding signature gives rise to an endofunctor  $\Sigma$  on the category  $\text{End}_{\text{Set}}$ .

A notion of model:  $\Sigma + \text{Id}_{\text{Set}}$ -algebra

The initial  $\Sigma + \text{Id}_{\text{Set}}$ -algebra of a binding signature  $\Sigma$  always exists.

Does this initial algebra come with a well-behaved substitution?



# Substitution [Fiore-Plotkin-Turi 1999]

The endofunctor  $\Sigma$  induced by a binding signature comes with a *strength* which allows [FPT] to refine the notion of model:

**$\Sigma$ -monoid:**

$\Sigma + \text{Id}_{\text{Set}}$ -algebra **equipped with a well-behaved substitution.**

**$\Sigma$ -monoid morphisms:**

algebra morphisms commuting with substitution.

**Theorem [FPT]:**

The initial  $\Sigma + \text{Id}_{\text{Set}}$ -algebra of a binding signature comes with a well-behaved substitution that makes it initial in the category of  **$\Sigma$ -monoids**.

This suggests defining signatures to be endofunctors on  $\text{End}_{\text{Set}}$  *with strength* (as in [Matthes-Uustalu 2004]).

# Table of contents

1. Binding signatures and their models
- 2. Signatures and models based on monads and modules**
3. Presentables signatures

# Our signatures and models

Binding signatures  $\hookrightarrow$  Endofunctors with strength  $\hookrightarrow$  Our signatures

A **signature**  $\Sigma$  is a functorial assignment:

$$R \mapsto \Sigma(R)$$

A **model of**  $\Sigma$  is a pair:  $(R, \rho : \Sigma(R) \rightarrow R)$

endofunctor with substitution

endofunctor with substitution

natural transformation preserving substitution

# Our signatures and models

Binding signatures  $\hookrightarrow$  Endofunctors with strength  $\hookrightarrow$  Our signatures

A **signature**  $\Sigma$  is a functorial assignment:

$$R \mapsto \Sigma(R)$$

A **model of**  $\Sigma$  is a pair:  $(R, \rho : \Sigma(R) \rightarrow R)$

monad  $:=$  endofunctor with substitution

module over a monad  $:=$  endofunctor with substitution

module morphism  $:=$  natural transformation preserving substitution

# Our signatures and models

Binding signatures  $\hookrightarrow$  Endofunctors with strength  $\hookrightarrow$  Our signatures

A **signature**  $\Sigma$  is a functorial assignment:

$$\begin{array}{c} \text{monad} \\ \searrow \\ R \mapsto \Sigma(R) \end{array}$$

A **model of**  $\Sigma$  is a pair:  $(R, \rho : \Sigma(R) \rightarrow R)$

monad := endofunctor with substitution

module over a monad := endofunctor with substitution

module morphism := natural transformation preserving substitution

# Our signatures and models

Binding signatures  $\hookrightarrow$  Endofunctors with strength  $\hookrightarrow$  Our signatures

A **signature**  $\Sigma$  is a functorial assignment:

$$\begin{array}{ccc} \text{monad} & & \text{module over } R \\ & \searrow & \swarrow \\ & R \mapsto \Sigma(R) & \end{array}$$

A **model of**  $\Sigma$  is a pair:  $(R, \rho : \Sigma(R) \rightarrow R)$

monad := endofunctor with substitution

module over a monad := endofunctor with substitution

module morphism := natural transformation preserving substitution

# Our signatures and models

Binding signatures  $\hookrightarrow$  Endofunctors with strength  $\hookrightarrow$  Our signatures

A **signature**  $\Sigma$  is a functorial assignment:

$$\begin{array}{ccc} \text{monad} & & \text{module over } R \\ & \searrow & \swarrow \\ & R \mapsto \Sigma(R) & \end{array}$$

A **model of**  $\Sigma$  is a pair:

$$(R, \rho : \Sigma(R) \rightarrow R)$$

monad  $\nearrow$

monad := endofunctor with substitution

module over a monad := endofunctor with substitution

module morphism := natural transformation preserving substitution

# Our signatures and models

Binding signatures  $\hookrightarrow$  Endofunctors with strength  $\hookrightarrow$  Our signatures

A **signature**  $\Sigma$  is a functorial assignment:

$$\begin{array}{ccc} \text{monad} & & \text{module over } R \\ & \searrow & \swarrow \\ & R \mapsto \Sigma(R) & \end{array}$$

A **model of**  $\Sigma$  is a pair:

$$\begin{array}{ccc} & (R, \rho : \Sigma(R) \rightarrow R) & \\ \nearrow & & \nwarrow \\ \text{monad} & & \text{module morphism} \end{array}$$

monad := endofunctor with substitution

module over a monad := endofunctor with substitution

module morphism := natural transformation preserving substitution



# Monads

## Reminder:

- $B(X)$  = expressions built out of  $0$ ,  $\star$  and variables taken in  $X$
- Variables induce a natural transformation  $\eta : \text{Id}_{\text{Set}} \rightarrow B$

It comes with a **substitution**  $\text{bind} : B(X) \rightarrow (X \rightarrow B(Y)) \rightarrow B(Y)$  required to satisfy some equations.

A triple  $(B, \eta, \text{bind})$  is called a **monad**.

A **monad morphism** between two monads  $R$  and  $S$  is a family of maps  $(f_X : R(X) \rightarrow S(X))_X$  preserving variables and substitution.

# Operations are module morphisms


In the  $(0, \star)$  language,

$$(t \star u)[x \mapsto v_x] = t[x \mapsto v_x] \star u[x \mapsto v_x]$$

**$\star$  commutes with substitution**

In the right hand side, substitution acts on a pair of expressions.

$B \times B$  supports  $B$ -substitution   $B \times B$  is a **module over  $B$**

$\star$  commutes with substitution   $\star : B \times B \rightarrow B$  is a **module morphism**

# Examples of modules over a monad

Some examples of **modules over a monad**  $R$ :

- $R$  itself
- $M \times N$  for any modules  $M$  and  $N$  (in particular,  $R \times R$ )
- The **derivative of a module**  $M$  is the module  $M'$  defined by  $M'(X) = M(X + \{x\})$ .

The derivative is used to model an operation binding a variable (Cf next slide).

# Examples of module morphisms

A **module morphism** between two modules  $M$  and  $N$  on the same monad  $R$  is a family of maps  $(f_x: M(X) \rightarrow N(X))_X$  commuting with substitution.

$$id_M : M \rightarrow M$$

the family of identity maps  $(id_{M(X)}: M(X) \rightarrow M(X))_X$  for any module  $M$

$$\star : B \times B \rightarrow B$$

$$app : L \times L \rightarrow L$$

the application operation of the lambda calculus monad  $L$ .

$$abs : L' \rightarrow L$$

Indeed, in  $\lambda x.t$ , the term  $t$  depends on an additional free variable  $x$ :

If  $t \in L(Y + \{x\}) = L'(\mathbf{Y})$ , then  $abs(t) = \lambda x.t \in L(Y)$

# Signatures and models

A **signature**  $\Sigma$  is a functorial assignment:

$$\begin{array}{ccc} \text{monad} & & \text{module over } R \\ & \searrow & \swarrow \\ & R \mapsto \Sigma(R) \end{array}$$

A **model of**  $\Sigma$  is a pair:

$$\begin{array}{ccc} & (R, \rho : \Sigma(R) \rightarrow R) & \\ \nearrow & & \nwarrow \\ \text{monad} & & \text{module morphism} \end{array}$$

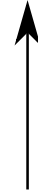
A **model morphism**  $m : (R, \rho) \rightarrow (S, \sigma)$  is a monad morphism commuting with the module morphism:

$$\begin{array}{ccc} \Sigma(R) & \xrightarrow{\rho} & R \\ \Sigma(m) \downarrow & & \downarrow m \\ \Sigma(S) & \xrightarrow{\sigma} & S \end{array}$$

# Existence of syntax?

Notion of signature too general: existence of the syntax (= **initial model**) ?

**Counter-example:** the signature  $R \mapsto \mathcal{P} \circ R$



powerset endofunctor on Set

# Examples of signatures with syntax

- $R \mapsto 1 + R \times R$

Models are monads  $R$  equipped with module morphisms  $1 \rightarrow R$  and  $R \times R \rightarrow R$ .

The syntax is the language  $B$  generated by a constant **0** and binary operation  $\star$ .

- $R \mapsto R \times R + R'$

Models are monads  $R$  equipped with two module morphisms:

$R \times R \rightarrow R$  and  $R' \rightarrow R$ .

The syntax is the lambda calculus.

# Algebraic signatures

More generally, the syntax exists for any signature induced by a disjoint sum of products of finite derivatives of the monad ( $R \mapsto R' \times R'' \times R''' + R \times R'' \times R''' \times R + \dots$ ).

We call such a signature an **algebraic signature**. They correspond to binding signatures through the inclusion:

Binding signatures  $\hookrightarrow$  Endofunctors with strength  $\xrightarrow{\mathcal{I}}$  Our signatures

**Our main result:** Quotients of algebraic signatures generate a syntax.



# Table of contents

1. Binding signatures and their models
2. Signatures and models based on monads and modules
- 3. Presentables signatures**

# Quotient of a signature

## Quotient of a set:

A quotient of a set  $X$  is a set  $Y$  together with a surjection  $p : X \rightarrow Y$ .

$$x \sim x' \iff p(x) = p(x')$$

## Quotient of a signature:

A quotient of a signature  $\Sigma$  is a signature  $\Psi$  together with a (natural) family of module morphisms  $(f_R : \Sigma(R) \rightarrow \Psi(R))_R$  that is pointwise surjective.

$$R \mapsto \begin{array}{c} \Sigma(R) \\ \downarrow f_R \\ \Psi(R) \end{array}$$

# Syntax for presentable signatures

A **presentable signature** is a quotient of an algebraic signature.

**Main Theorem:** For any presentable signature, there is a syntax.

We now give examples of new kinds of operations specified by presentable signatures (more can be found in the article).

# Example 1: Symmetric operations

**Binary commutative operation  $+$ :**

$$t + u = u + t$$

As a quotient of an algebraic signature:

$$R \mapsto \begin{array}{c} R \times R \\ \downarrow \\ R \times R / \{(x, y) \sim (y, x)\} \end{array}$$

This generalizes to **n-ary permutation invariant operations**.

## Example 2: Explicit substitution

An operation  $\_ \langle x_i \mapsto t_i \rangle$  that mimics the behavior of the substitution in the sense that it enjoys some of its coherences:

- invariance under **permutation**

$$F(x, y) \langle x \mapsto t, y \mapsto u \rangle = F(y, x) \langle x \mapsto u, y \mapsto t \rangle$$

- invariance under **weakening**

$$F(x) \langle x \mapsto t, y \mapsto u \rangle = F(x) \langle x \mapsto u \rangle$$

- invariance under **contraction**

$$F(x, y) \langle x, y \mapsto t \rangle = F(x, x) \langle x \mapsto t \rangle$$

# Example 2: Explicit substitution

Explicit substitution as a quotient of the algebraic signature:

$$\begin{array}{c}
 \Sigma(R) := R' \times R \quad + \quad R'' \times R \times R \quad + \quad R''' \times R \times R \times R \quad + \quad \dots \\
 \begin{array}{ccc}
 \nearrow & \uparrow & \nwarrow \\
 t\langle x \mapsto u \rangle & t\langle x \mapsto u, y \mapsto v \rangle & t\langle x \mapsto u, y \mapsto v, z \mapsto w \rangle
 \end{array} \\
 \\
 \begin{array}{c}
 \Sigma(R) \\
 \downarrow \\
 \Sigma(R) / \sim
 \end{array}
 \end{array}$$

$R \mapsto$

- **permutation:**  $t\langle x \mapsto u, y \mapsto v \rangle \sim t[x \rightleftharpoons y]\langle x \mapsto v, y \mapsto u \rangle$
- **weakening:**  $t\langle x \mapsto u \rangle \sim t\langle x \mapsto u, y \mapsto v \rangle$
- **contraction:**  $t\langle x \mapsto u, y \mapsto u \rangle \sim t[y := x]\langle x \mapsto u \rangle$

# Conclusion

We have given a criterion for signatures to generate a syntax. This criterion encompasses the classical binding signatures. It also allows new operations that satisfy some equations.

Our main theorem has been formalized using the Coq library UniMath.

## **Future work:**

- extend our framework to encompass general equations (e.g. associative binary operation,  $\lambda$ -calculus modulo  $\beta/\eta$  equivalence);
- extend our framework to simply typed syntaxes.

# Conclusion

We have given a criterion for signatures to generate a syntax. This criterion encompasses the classical binding signatures. It also allows new operations that satisfy some equations.

Our main theorem has been formalized using the Coq library UniMath.

## **Future work:**

- extend our framework to encompass general equations (e.g. associative binary operation,  $\lambda$ -calculus modulo  $\beta/\eta$  equivalence);
- extend our framework to simply typed syntaxes.

Thank you!