

High-level signatures and initial semantics

Ambroise Lafont

joint work with Benedikt Ahrens, André Hirschowitz, Marco Maggesi

CSL 2018

Overview

Topic: specification and construction of untyped syntaxes with variables and a well-behaved substitution (e.g. lambda calculus).

Our work:

1. general notion of **signature** based on **monads** and **modules**.
 - *Caveat:* Not all of them do **generate a syntax**
 - special case: classical **algebraic signatures** generate a syntax
2. our main result: any **quotient** of algebraic signatures generates a syntax.

This talk: explain the words in bold

Operations covered by our result

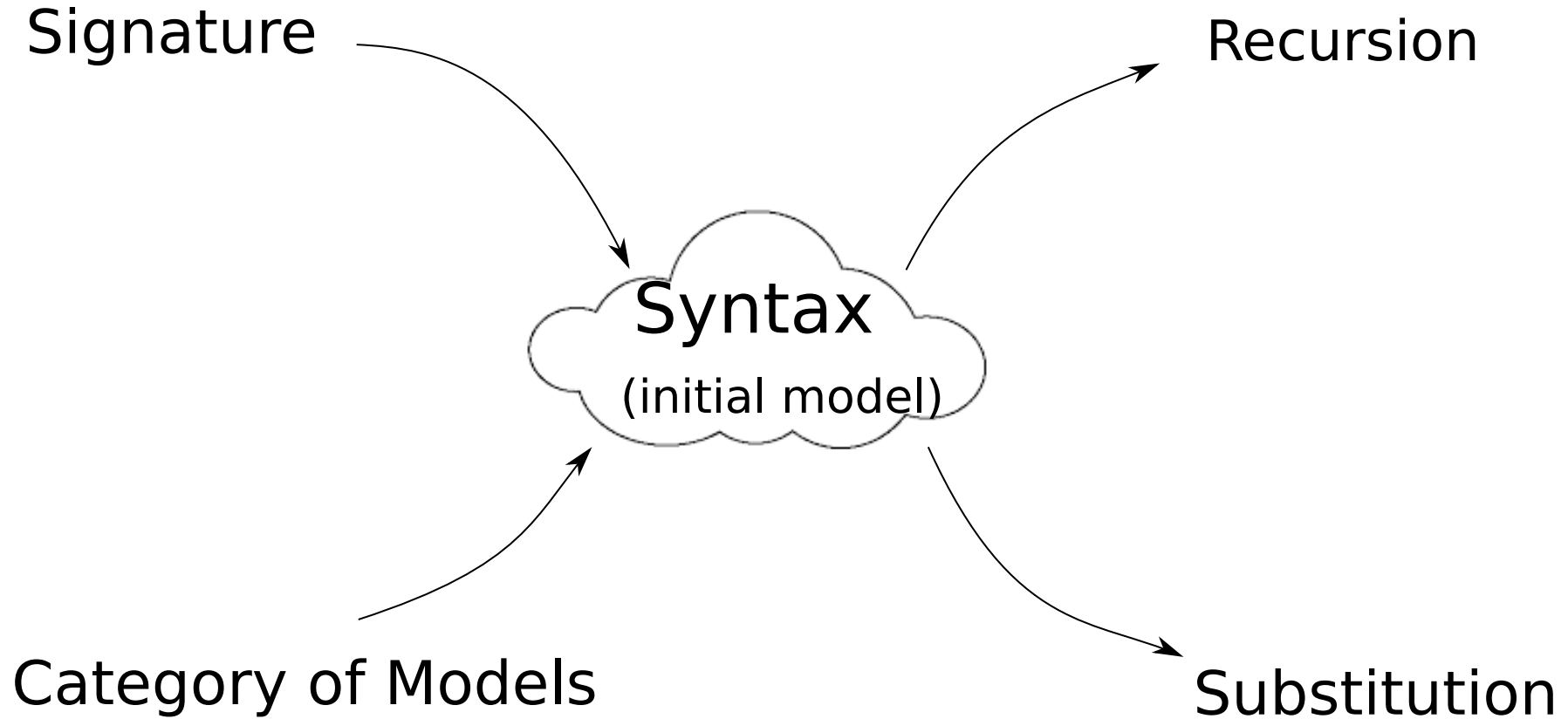
Some examples:

- Symmetric operations

$$m : T \times T \rightarrow T \quad \text{s.t.} \quad m(t, u) = m(u, t)$$

- Explicit substitution with coherences
- Fixed point operation with coherences
- Syntactic closure operator with coherences

What is a syntax?



generates a syntax = existence of the initial model

Table of contents

- 1. Review: Binding signatures and their models**
2. Signatures and models based on monads and modules
3. Our main result

Table of contents

1. Review: Binding signatures and their models

- Categorical formulation of term languages
- Initial semantics for binding signatures

2. Signatures and models based on monads and modules

3. Our main result

Categorical formulation of a term language

Example: syntax with a binary operation \star , 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):

$$B : X \mapsto \{\text{expressions over } X\}$$

For example:

$$\begin{aligned} B(\emptyset) &= \{0, 0 \star 0, \dots\} \\ B(\{x, y\}) &= \{0, 0 \star 0, \dots, x, y, x \star y, \dots\} \end{aligned}$$

Categorical formulation of a term language

Then we have:

$$\star : B \times B \rightrightarrows B$$

$$0 : 1 \rightrightarrows B$$

$$\text{var} : \text{Id}_{\text{Set}} \rightrightarrows B$$

Putting all together:

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

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

Actually, B can be **characterized** as the initial algebra.

Binding Signatures

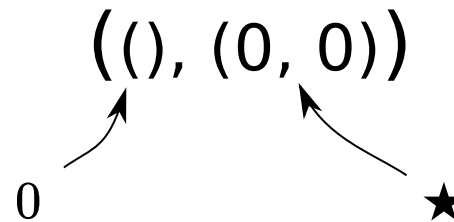
Definition

Binding signature = a family of lists of natural numbers.

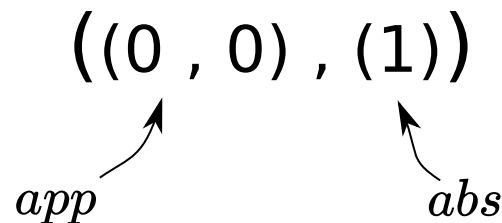
Each list specifies one operation in the syntax:

- 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:



Initial semantics for binding signatures

Reminder

The syntax $(0, \star)$ is the initial algebra for the endofunctor:

$$F \mapsto F \times F + 1 + \text{Id}_{\text{Set}}$$

More generally, any binding signature gives rise to an endofunctor Σ .

Definition

Model = $(\Sigma + \text{Id}_{\text{Set}})$ -algebra

Classical Theorem

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

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

Answer: Yes: see e.g. [Fiore, Plotkin, Turi 1999], [Ghani & Uustalu 2003]

Table of contents

1. Review: Binding signatures and their models

2. Signatures and models based on monads and modules

- Our take on substitution
- Our take on signatures, models and syntax
- Our take on binding signatures

3. Our main result

The Big Picture of signatures and models

Binding signatures \hookrightarrow Our signatures

A **signature** Σ is a functorial assignment:

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

A **model of** Σ 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

The Big Picture of signatures and models

Binding signatures \hookrightarrow Our signatures

A **signature** Σ is a functorial assignment:

$$\text{monad} \curvearrowright R \mapsto \Sigma(R)$$

A **model of** Σ 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

The Big Picture of signatures and models

Binding signatures \hookrightarrow Our signatures

A **signature** Σ is a functorial assignment:

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

monad \nearrow \nwarrow module over R

A **model of** Σ 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

The Big Picture of signatures and models

Binding signatures \hookrightarrow Our signatures

A **signature** Σ is a functorial assignment:

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

monad \nearrow \nwarrow module over R

A **model of** Σ 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

The Big Picture of signatures and models

Binding signatures \hookrightarrow Our signatures

A **signature** Σ is a functorial assignment:

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

monad \nearrow \nwarrow module over R

A **model of** Σ is a pair:

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

monad \nearrow \nwarrow module morphism

monad := endofunctor with substitution

module over a monad := endofunctor with substitution

module morphism := natural transformation preserving substitution

Substitution and monads

Reminder:

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

Substitution:

$$\text{bind} : B(X) \rightarrow (X \rightarrow B(Y)) \rightarrow B(Y)$$

+ laws

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

monad morphism = mapping preserving var and bind .

Preview: Operations are module morphisms

★ commutes with substitution

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

Categorical formulation

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

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

Building blocks for binding signatures

Essential constructions 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 $M'(X) = M(X + \{\bullet\})$.

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

Syntactic operations are module morphisms

module morphism = maps commuting with substitution.

$$id_M : M \rightarrow M$$

$$0 : 1 \rightarrow B$$

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

$$app : \Lambda \times \Lambda \rightarrow \Lambda$$

$$abs : \Lambda' \rightarrow \Lambda$$

The Big Picture again

A **signature** Σ is a functorial assignment:

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

monad \quad module over R

A **model of** Σ is a pair:

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

monad \quad module morphism

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

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

Syntax

Definition

Given a signature Σ , its **syntax** is an initial object in its category of models.

Question: Does the syntax exist for every signature?

Answer: No.

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



powerset endofunctor on Set

Examples of signatures generating syntax

- **(0,★) language:**

Signature: $R \mapsto 1 + R \times R$

Model: $(R, \quad 0 : 1 \rightarrow R, \quad \star : R \times R \rightarrow R)$

Syntax: $(B, \quad 0 : 1 \rightarrow B, \quad \star : B \times B \rightarrow B)$

- **lambda calculus:**

Signature: $R \mapsto R' + R \times R$

Model: $(R, \quad abs : R' \rightarrow R, \quad app : R \times R \rightarrow R)$

Syntax: $(\Lambda, \quad abs : \Lambda' \rightarrow \Lambda, \quad app : \Lambda \times \Lambda \rightarrow \Lambda)$

Can we generalize this pattern?

Initial semantics for algebraic signatures

Theorem [Hirschowitz & Maggesi 2007]

Syntax exists for any **algebraic signature**, i.e. signature built out of derivatives, products, coproducts, and the trivial signature $R \mapsto R$.

Algebraic signatures correspond to binding signatures through the embedding:

Binding signatures \hookrightarrow Our signatures

Question: Can we identify a larger class of signatures generating a syntax?

Table of contents

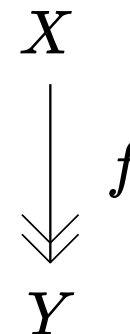
1. Review: Binding signatures and their models
2. Signatures and models based on monads and modules
- 3. Our main result**
 - Definition of presentable signatures
 - Generated syntax for presentable signatures
 - Examples of presentable signatures

Quotient of a signature

Quotient of a set:

A quotient of a set X consists of:

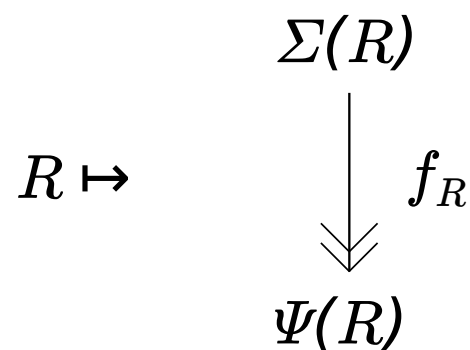
- a set Y
- a surjective function $f: X \rightarrow Y$



Quotient of a signature:

A quotient of a signature Σ consists of:

- a signature Ψ
- a (natural) family of surjective module morphisms $(f_R: \Sigma(R) \rightarrow \Psi(R))_R$



Syntax for presentable signatures

Definition

A **presentable signature** Ψ is a quotient of an algebraic signature Σ :

$$\begin{array}{c} \Sigma \\ \downarrow \\ \Psi \end{array}$$

Main Theorem

Any presentable signature generates a syntax.

Question: Are there interesting examples of presentable signatures?

Answer:

- Symmetric operations
- Explicit substitution with coherences
- Fixed point operation with coherences
- Syntactic closure operator with coherences

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 $_ \langle x_i \mapsto t_i \rangle$
- satisfying coherence equations:
 - 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**

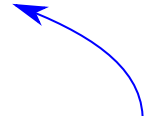
$$F(x, y) \langle x, y \mapsto t \rangle = F(x, x) \langle x \mapsto t \rangle$$

Example 2: Explicit substitution

$$\Sigma(R) := R' \times R + R'' \times R \times R + R''' \times R \times R \times R + \dots$$

$$t\langle x \mapsto u \rangle$$


$$t\langle x \mapsto u, y \mapsto v \rangle$$


$$t\langle x \mapsto u, y \mapsto v, z \mapsto w \rangle$$


Signature of explicit substitution as a quotient of an algebraic signature:

$$\begin{array}{c} \Sigma(R) \\ \downarrow \\ R \mapsto \\ \Sigma(R) / \sim \end{array}$$

- **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

Summary of the talk:

- presented a notion of signature and models
- identified a class of signatures that generate a syntax
 - encompasses the classical binding signatures
 - encompasses operations satisfying some equations

The main theorem has been formalized in Coq using the UniMath library.

Future work:

- add equations (e.g. lambda calculus modulo beta/eta equivalence);
- extend our framework to simply typed syntaxes.

Conclusion

Summary of the talk:

- presented a notion of signature and models
- identified a class of signatures that generate a syntax
 - encompasses the classical binding signatures
 - encompasses operations satisfying some equations

The main theorem has been formalized in Coq using the UniMath library.

Future work:

- add equations (e.g. lambda calculus modulo beta/eta equivalence);
- extend our framework to simply typed syntaxes.

Thank you!