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***.
    a. Caveat: Not all of them do ***generate a syntax***
    b. covers the classical **binding signatures**
2. our main result: any ***quotient*** of algebraic signatures also generates a syntax

This talk: explain the words in bold.

Operations covered by our result

Some examples:

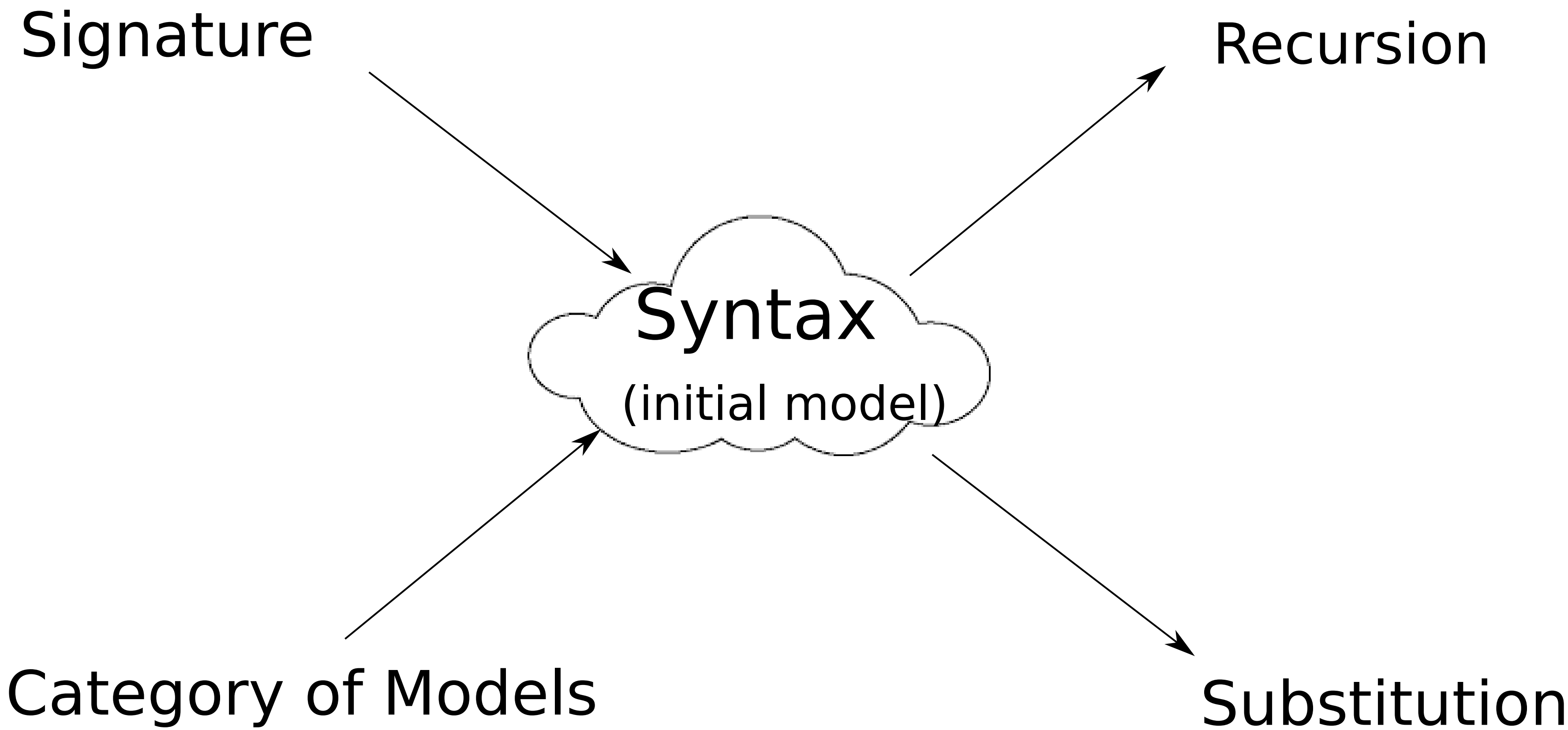
- Symmetric operations

- Explicit substitution

- Coherent fixed point operation

- Syntactic closure operator

What is a syntax?



generates a syntax = existence of the initial model

Table of contents

Categorical account of a term language

$$\begin{aligned} \mathrm{expr} ::= {} & x && \textit{(variable)} \\ \mid {} & t_1 \bigstar t_2 && \textit{(binary operation)} \\ \mid {} & 0 && \textit{(constant)} \end{aligned}$$

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

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

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

Categorical account of a term language

The binary operation ★ induces a natural transformation:

$$B \times B \to B$$

The constant **0** induces a natural transformation:

$$1 \to B$$

Variables induce a natural transformation

$$\mathrm{Id}_{\mathrm{Set}} \to B$$

They gather into a single natural transformation:

$$B \times B + 1 + \mathrm{Id}_{\mathrm{Set}} \to B$$

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

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, ★:

$$\big(\underset{0}{()},\ \underset{\star}{(0,\ 0)}\big)$$

Lambda calculus:

$$\big(\underset{app}{(0\ ,\ 0)}\ ,\ \underset{abs}{(1)}\big)$$

Signatures and endofunctors

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

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

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

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

Substitution [Fiore-Plotkin-Turi 1999]

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

$\boldsymbol{\Sigma}$-monoid:

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

$\boldsymbol{\Sigma}$-monoid morphisms:

algebra morphisms commuting with substitution.

Theorem [FPT]:

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

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

Table of contents

The Big Picture

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

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

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

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

$$\begin{aligned} \text{monad} &:= \text{endofunctor with substitution} \\ \text{module over a monad} &:= \text{endofunctor with substitution} \\ \text{module morphism} &:= \text{natural transformation preserving substitution} \end{aligned}$$

The Big Picture

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

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

monad

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

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

monad	:=	endofunctor with substitution
module over a monad	:=	endofunctor with substitution
module morphism	:=	natural transformation preserving substitution

The Big Picture

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

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

$$\underbrace{R}_{\text{monad}} \mapsto \underbrace{\Sigma(R)}_{\text{module over } R}$$

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

monad	:=	endofunctor with substitution
module over a monad	:=	endofunctor with substitution
module morphism	:=	natural transformation preserving substitution

The Big Picture

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

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

$$\underbrace{R}_{\text{monad}} \mapsto \underbrace{\Sigma(R)}_{\text{module over } R}$$

A **model of Σ** is a pair:

$$(\underbrace{R}_{\text{monad}}, \quad \rho : \Sigma(R) \to R)$$

monad	:=	endofunctor with substitution
module over a monad	:=	endofunctor with substitution
module morphism	:=	natural transformation preserving substitution

The Big Picture

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

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

monad　　　　module over R

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

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

monad　　　　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, ★ and variables taken in X
- Variables induce a natural transformation $\eta : \mathrm{Id}_{\mathrm{Set}} \to B$

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

A triple (B, η, bind) is called a **monad**.

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

Preview: Operations are module morphisms

$\star$ commutes with substitution

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

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

Categorical formulation

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

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

Building blocks for binding signatures

Some examples of **modules over a monad $\boldsymbol{R}$**:

- R itself
- M x N for any modules M and N (in particular, R x R)
- The **derivative of a module $\boldsymbol{M}$** 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).

Syntactic operations are module morphisms

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

$id_M : M \to M$

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

$\star : B \times B \to B$

$app : L \times L \to L$

the application operation of the lambda calculus monad L.

$abs : L' \to L$

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

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

The Big Picture again

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

monad module over R

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

A **model of** $\boldsymbol{\Sigma}$ is a pair:

$$(R, \quad \rho : \Sigma(R) \to R)$$

monad module morphism

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

$$\begin{tikzcd} \Sigma(R) \arrow[r, "\rho"] \arrow[d, "\Sigma(m)"'] & R \arrow[d, "m"] \\ \Sigma(S) \arrow[r, "\sigma"'] & S \end{tikzcd}$$

Syntax

Definition

A **syntax** is an initial object in the category of models of a signature

Notion of signature too general: existence of the syntax ?

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

powerset endofunctor on $\mathbb{S}\mathrm{et}$

Examples of signatures with syntax

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

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

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

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

  Models are monads R equipped with two modules morphisms: $R \times R \to R$ and $R' \to R$.

  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 ($R \mapsto R' \times R'' \times R''' + R \times R'' \times R''' \times R + \ldots$).

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

$$\text{Binding signatures} \hookrightarrow \text{Endofunctors with strength} \hookrightarrow \text{Our signatures}$$

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. Our main result

- Presentable signatures
- Generated syntax for presentable signatures
- Examples of application

Quotient of a signature

Quotient of a set:

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

$$x \sim x' \qquad \Longleftrightarrow \qquad p(x) = p(x')$$

Quotient of a signature:

A quotient of a signature Σ consists of:

- a signature Ψ

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

$$R \mapsto \begin{tikzcd} \Sigma(R) \arrow[d, twoheadrightarrow, "f_R"] \\ \Psi(R) \end{tikzcd}$$

Syntax for presentable signatures

Definition

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

Theorem

Any presentable signature generates a syntax.

Examples follow.

Example 1: Symmetric operations

Binary commutative operation +:

$$t + u = u + t$$

As a quotient of an algebraic signature:

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

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:

$$\Sigma(R) := R' \times R \; + \; R'' \times R \times R \; + \; R''' \times R \times R \times R \; + \; \ldots$$

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

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

- **permutation**: $t\langle x \mapsto u, y \mapsto v \rangle \sim t[x \rightleftarrows y]\langle x \mapsto v, y \mapsto u \rangle$
- **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:

- criterion for signatures to generate a syntax
- encompasses the classical binding signatures
- new operations that satisfy some equations
- formalized using the Coq library UniMath

Future work:

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

Conclusion

Summary:

- criterion for signatures to generate a syntax

- encompasses the classical binding signatures

- new operations that satisfy some equations

- formalized using the Coq library UniMath

Future work:

- add equations (e.g. lambda calculus modulo beta/eta equivalence);

- extend our framework to simply typed syntaxes.

Thank you!