High-level signatures and initial semantics

Ambroise Lafont

joint work with Benedikt Ahrens, André Hirschowitz, Marco Maggesi

CSL 2018

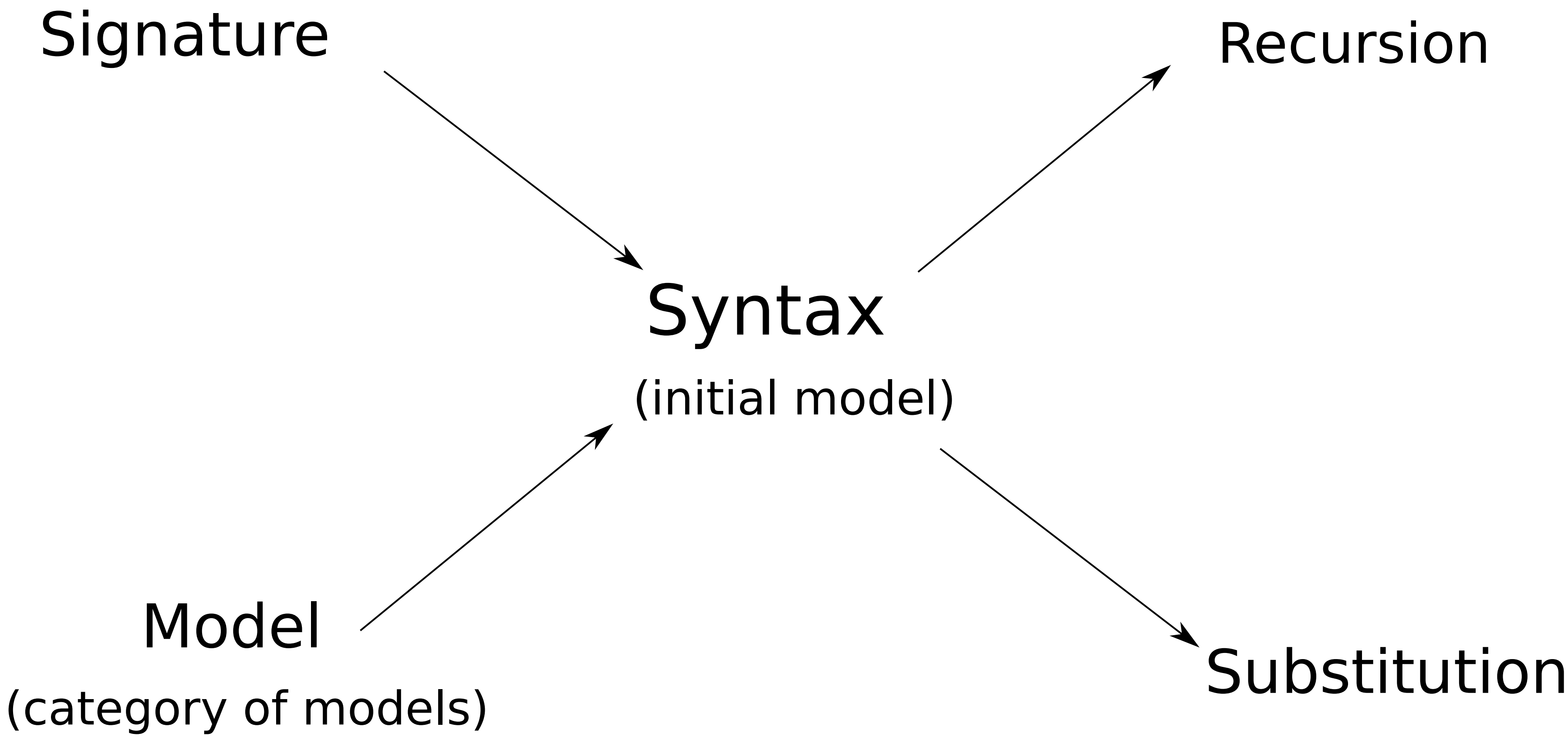
Overview

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

Example: 0, ★

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

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

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

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

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

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

Example: 0, ★

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, ⋆:

$$\big((), (0, 0)\big)$$

nullary operation **0**   binary operation ⋆

Lambda calculus:

$$\big((0\ ,\ 0)\ ,\ (1)\big)$$

application   lambda-abstraction

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

Our signatures and models

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

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

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

A **model of $\boldsymbol{\Sigma}$** 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

Our signatures and models

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

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

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

Our signatures and models

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	:=	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** 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

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** Σ is a functorial assignment:

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

A **model of Σ** is a pair:

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

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, ★ 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.

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 $\rightsquigarrow$ $B \times B$ is a **module over B**

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

Examples of modules over a monad

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).

Examples of module morphisms

A **module morphism** between two modules M and N on the same monad R is a family of maps $(f_X{:}M(X) \to N(X))_X$ 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 $t \in L(Y + \{x\})$ **= $L'(Y)$**, then $abs(t) = \lambda x.t \in L(Y)$

Signatures and models

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

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

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

Existence of syntax?

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

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 $\star$.

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

  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:

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

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

Table of contents

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

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

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:

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

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:

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

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

$$R \mapsto \begin{tikzcd} \Sigma(R) \arrow[d, two heads] \\ \Sigma(R)\,/\sim \end{tikzcd}$$

- **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

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, lambdacalculus 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, lambdacalculus modulo beta/eta equivalence);

- extend our framework to simply typed syntaxes.

Thank you!