High-level signatures and initial semantics

Ambroise Lafont

joint work with Benedikt Ahrens, André Hirschowitz, Marco Maggesi

CSL 2018

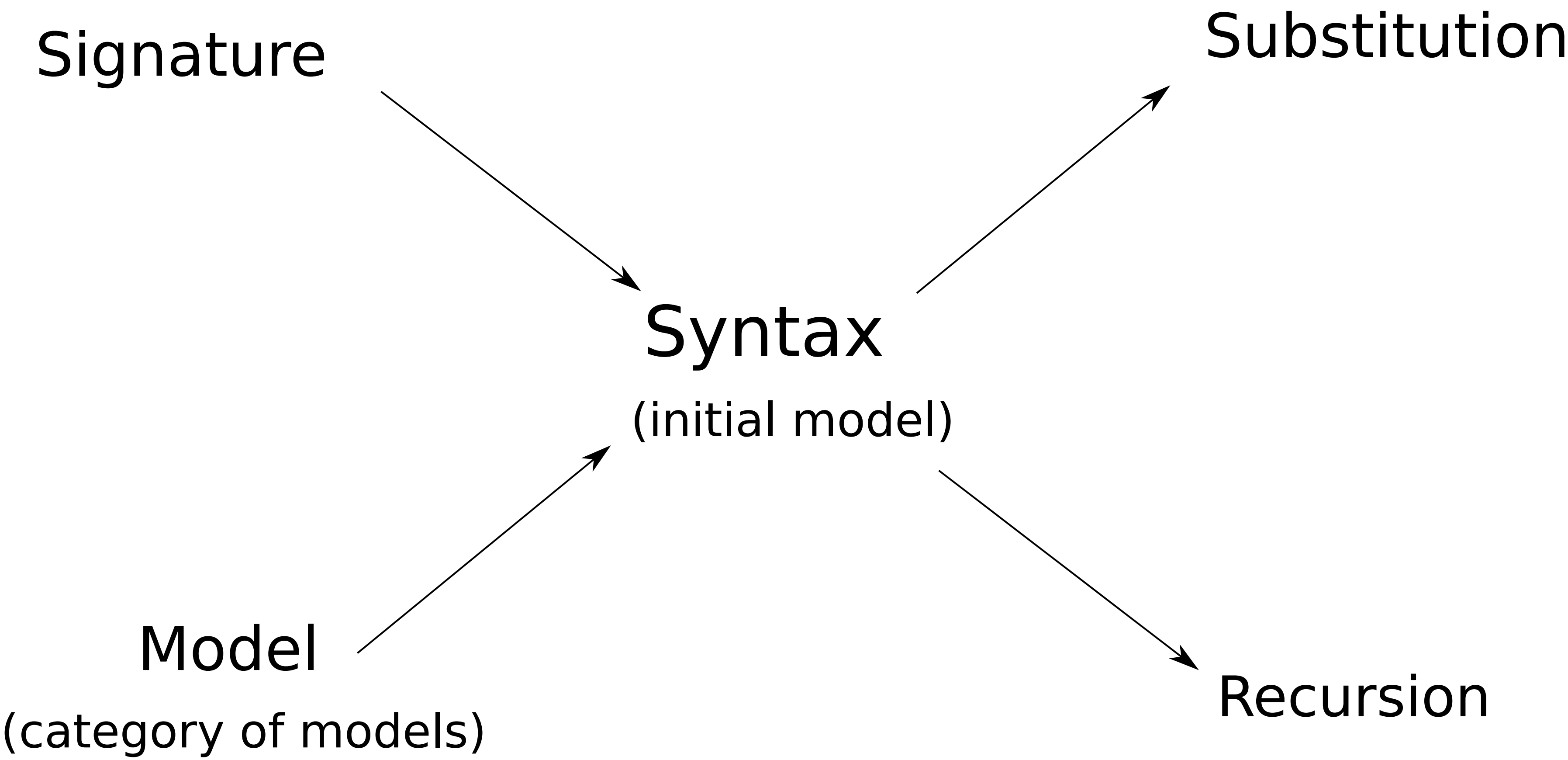
Introduction

Purpose of our work: 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 have a notion of *signature* too general (all of them do not *specify a syntax*)
2. classical *binding signatures* embed into our signatures as *algebraic signatures,* and indeed specify a syntax.
3. our main result: any *quotient* of algebraic signatures also specifies a syntax

What is a syntax?



Signatures which we care about: those whose category of models have an *initial object*.

Our work

We present a notion of signature (and associated models) based on the notion of module over a monad.

Goal of our work: Identify a large class of these signatures whose category of models have an initial object.

Our main result: Quotients of "binding signatures" have a syntax.

Table of contents

Example: 0, ★

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

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

The syntax induces an endofunctor B (on Set) mapping a set of variables to the set of expressions built out of them:

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

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

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

Using disjoint union, 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 $\mathrm{End}_{\mathrm{Set}}$ of endofunctors on Set.

Actually, B can be defined to be the initial algebra of $\boldsymbol{F}$.

Binding Signatures [Fiore-Plotkin-Turi 1999]

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, ★: $\big((), (0, 0)\big)$

0-aire operation **0** binary operation ★

Lambda calculus: $\big((0\ ,\ 0)\ ,\ (1)\big)$

application lambda-abstraction

Signatures and endofunctors [FPT]

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

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

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

What about substitution?

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

Substitution following [FPT]

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

***Σ*-monoids =** $\Sigma + \mathrm{Id}_{\mathrm{Set}}$-algebras equipped with a well-behaved substitution.

***Σ*-monoid morphisms** = algebra morphisms commuting with substitution.

Theorem [FPT]:

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

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

Our signatures

In the next slides, we present our notion of signature.

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

Conjecture: for any endofunctor with strength Σ, our category of models is equivalent to the [FPT] one:

$$\mathrm{Models}(\mathcal{I}(\Sigma)) \cong \Sigma\text{-monoids}$$

(modulo a technical restriction, namely considering only finitary endofunctors on Set)

Our 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:

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

monad   module morphism

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

Table of contents

Monads

The functor B corresponds to the language (0, ★) with variables as placeholders for any expression of B.

$B(X)$ denotes the set of expressions taking variables in X.

Substitution (required to satisfy some intuitive equations):

$$B(X) \to (X \to B(Y)) \to B(Y)$$

Such a functor 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 as module morphisms

In the (0, ★) language,

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

★ commutes with substitution

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

We abstract this situation as follows:

- pairs of expressions form a **module** $B \times B$ over the monad B,

- ★ yields **module morphism** from $B \times B$ to B

Module over a monad

The endofunctor $B \times B$ corresponds to expressions with variables as placeholders for any expression in the language B.

Substitution with B-expressions (required to satisfy some intuitive equations):

$$(B \times B)(X) \to (X \to B(Y)) \to (B \times B)(Y)$$

Such a functor is called a **module over the monad $\boldsymbol{B}$**.

Examples of modules

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)

- M' is the module defined by M'(X) = M(X + {x}) for any set X of variables given a module M. We call it the **derivative of $\boldsymbol{M}$**.

  The new variable x is used to model an operation binding a variable (e.g. the lambda-abstraction).

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

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:

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

monad module morphism

A **model morphism** $m : R \to S$ is a monad morphism commuting with σ:

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

Existence of syntax = initial model?

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

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

  By universal property of the disjoint sum, models are monads R equipped with module morphisms $1 \to R$ and $R \times R \to R$. The syntax corresponds to our example with **0** and ★.

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

  Models are monads R equipped with two modules morphisms: $R \times R \to R$ and $R' \to R$. The syntax corresponds to lambda calculus.

Algebraic signatures

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

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

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

Our main result: Quotients of algebraic signatures have 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.

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

Presentable signatures

A **presentable signature** is a quotient of a binding 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 $\_\langle \mathbf{x_i} \mapsto \mathbf{t_i} \rangle$ that mimics the behavior of the meta-substitution $\_[\mathbf{x_i} \mapsto \mathbf{t_i}]$ 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:

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

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

Quotiented by the following relation:

- **permutation**: $t\langle x \mapsto u, y \mapsto v \rangle \sim t[x \rightleftarrows y]\langle x \mapsto v, y \mapsto u \rangle$
- **weakening**: $t\langle x \mapsto u \rangle \sim t\langle x \mapsto v, y \mapsto u \rangle$
- **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 specify a syntax. This criterion encompasses the classical binding signatures, and allows new operations in the syntax.

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

Future work:

- take into account more sophisticated equations in the syntax than just quotients (e.g. associative binary operation, lambda-calculus modulo beta/eta equivalence);

- extend our framework to simply typed syntaxes.

FIN PROVISOIRE

Ne pas lire les slides qui suivent (ce sont des anciennes slides que je garde au cas où).

FIN PROVISOIRE

Ne pas lire les slides qui suivent (ce sont des anciennes slides que je garde au cas où).

Fixpoint operator with coherences

But we would like to encode some of the expected behaviour of such a fixed point:

- invariance under permutation
- invariance under weakening
- invariance under contraction. Roughly:

```
let rec f₁ = F(f₁,f₂)
    and f₂ = F(f₁,f₂)
in f₁
```

=

```
let rec f = F(f,f)
in f
```

A construction satisfying these invariances can be specified by quotienting the naive algebraic signature.

Fixpoint operator

A fixpoint operator:

A language with (mutual) fixpoints comes with a construction

$$\begin{aligned}&\texttt{let rec } \mathbf{f}_1 = \mathbf{t}_1\\ &\qquad \texttt{and } \mathbf{f}_2 = \mathbf{t}_2\\ &\qquad\qquad \cdots\\ &\qquad \texttt{and } \mathbf{f}_n = \mathbf{t}_n\\ &\texttt{in } \mathbf{f}_i\end{aligned}$$

where each $\mathbf{f}_j$ may appear as a variable in each expression $\mathbf{t}_i$.

Thus, it takes **n** expressions $\mathbf{t}_1,..,\mathbf{t}_n$ depending on **n** fresh variables $\mathbf{f}_1,..,\mathbf{f}_n$ and produces an expression which no longer depend on them.

As such, it can be specified by a binding signature.

Example 2: Syntactic closure operator

Syntactic closure operator $\forall$

$\forall xyz.t$ binds the variables x, y and z in the term t

Example of an operation invariant under **permutation** and **weakening**:

- permutation: $\forall xy.t = \forall yx.t$

- weakening: $\forall x.t = \forall xy.t$ if t does not depend on y