Signatures and models for syntax and operational semantics in the presence of variable binding

Ambroise LAFONT[1]

[1]DAPI
IMT Atlantique

PhD, 2019

Outline

Outline

Ingredients

- Programming languages (PLs) as graphs
  - (**Syntax**) vertices = terms
  - (**Semantics**) arrows = reductions between terms
- Parallel substitution: variables $\mapsto$ terms
  - monads and modules over them
- (untyped PLs)

Example

λ-calculus with β-reduction:

- **Syntax:** $S, T \quad ::= x | S\ T | \lambda x.S$
- **Reductions:** $(\lambda x.t)\, u \xrightarrow{\beta} t[x \mapsto u] \quad + \quad$ congruences

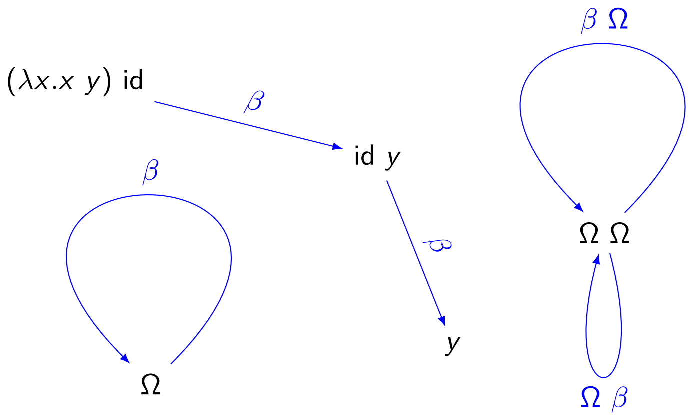
modulo α-equivalence, e.g.

$$\lambda x.x = \lambda y.y$$

Outline

PLs as graphs

Example: λ-calculus with β-reduction



- (**Syntax**) vertices = terms
- (**Semantics**) arrows = reductions (dedicated syntax: Cf labels)

Graphs

Definition

Graph = a quadruple (A, V, σ, τ) where

$$A \overset{\sigma}{\underset{\tau}{\rightrightarrows}} V$$

$A = \{\text{arrows}\}$ $\qquad$ $V = \{\text{vertices}\}$

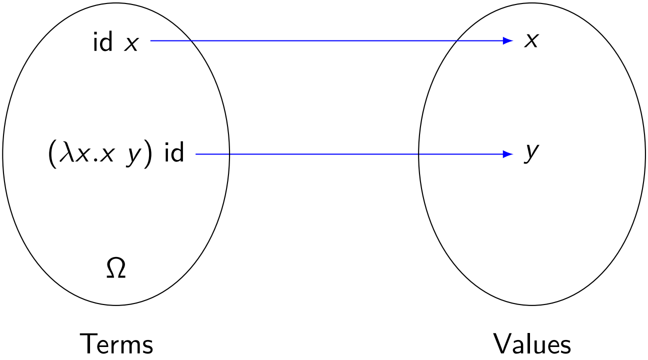
$$\begin{array}{rccc} \sigma : & A & \to & V \\ & t \xrightarrow{r} u & \mapsto & t \end{array} \qquad \begin{array}{rccc} \tau : & A & \to & V \\ & t \xrightarrow{r} u & \mapsto & u \end{array}$$

$$\sigma(r) \xrightarrow{r} \tau(r)$$

PLs as bipartite graphs

Example: λ-calculus cbv with big-step operational semantics

- term $\to$ value
- variables = placeholders for values



Bipartite graphs
Definition

Bipartite graph $=$ a quadruple (A, V_1, V_2, ∂) where

$$V_1 \xleftarrow{\sigma} A \xrightarrow{\tau} V_2$$

$$A = \{\text{arrows}\} \qquad \begin{aligned} V_1 &= \{\text{vertices in first group}\} \\ V_2 &= \{\text{vertices in second group}\} \end{aligned}$$

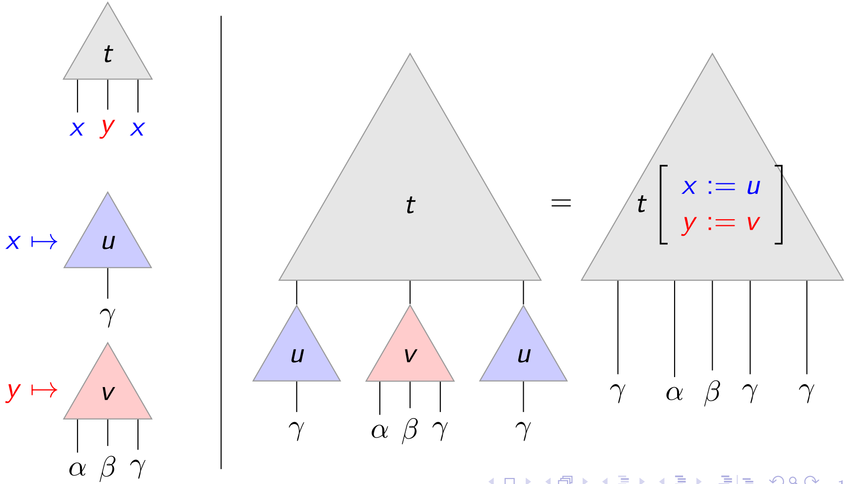
For simplicity, we focus on the particular case of **graphs**: $V_1 = V_2$.

Outline

Parallel substitution

Syntax comes with substitution

terms (e.g. λ-terms) = trees with free variables as (distinguished) leaves.



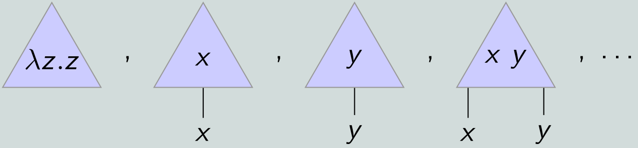
Parallel substitution made formal

Free variables indexing

$$X \mapsto \{\text{terms taking free variables in } X\}$$

Example: λ-calculus

$L(\{x, y\}) = \{$



$\}$

Parallel substitution

For any $f : X \to L(Y)$,

$$\begin{array}{rccc} \mathrm{bind}_f : & L(X) & \to & L(Y) \\ & t & \mapsto & t[x \mapsto f(x)] \qquad (\text{or } t[f]) \end{array}$$

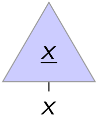
Monads

λ-calculus as a monad (L, bind, η)

1. Parallel substitution (L, bind)
2. Variables are terms

$$\eta_X : \quad X \quad \to \quad L(X)$$

$$x \quad \mapsto$$



3. Monadics laws:

$$\underline{x}[f] = f(x) \qquad\qquad t[x \mapsto \underline{x}] = t$$

$+$ associativity:

$$t[f][g] = t\,[x \mapsto f(x)[g]]$$

Substitution for semantics

Our notion of PL:

- **Syntax**: a monad (L, bind, η)
- **Semantics**:
  - graphs $R(X) \underset{\tau}{\overset{\sigma}{\rightrightarrows}} L(X)$ for each X

$$R(X) = \begin{array}{c}\text{total set of reductions between} \\ \text{terms taking free variables in } X\end{array}$$

  - substitution of reduction: variables $\mapsto$ L-**terms**.

$$\frac{t \xrightarrow{r} u}{t[f] \xrightarrow{r[f]} u[f]}$$

Substitution for semantics made formal

R as a **module** over L

For any $f : X \to \boldsymbol{L}(Y)$,

$$\begin{aligned} \mathrm{bind}_f : \quad R(X) &\to R(Y) \\ r &\mapsto r[x \mapsto f(x)] \quad (\text{or } r[f]) \end{aligned}$$

s.t.

$$r[x \mapsto \underline{x}] = r \qquad\qquad r[f][g] = r\,[x \mapsto f(x)[g]]$$

σ and τ as L-**module morphisms**

$$\sigma(r[f]) \xrightarrow{r[f]} \tau(r[f])$$

Then, $\dfrac{\sigma(r) \xrightarrow{r} \tau(r)}{\sigma(r)[f] \xrightarrow{r[f]} \tau(r)[f]}$ enforces $\begin{array}{l} \sigma(r[f]) = \sigma(r)[f] \\ \tau(r[f]) = \sigma(r)[f] \end{array}$

Commutation with substitution $\Leftrightarrow$ Module morphisms $\sigma, \tau : R \to L$.

Reduction monads

Definition

A **reduction monad** is a quadruple $R \overset{\sigma}{\underset{\tau}{\rightrightarrows}} T$ s.t.

- T = monad
- R = module over T
- $\sigma, \tau : R \to T$ are T-module morphisms.

Example

λ-calculus with β-reduction.

How can we specify a reduction monad?

1. signature for the (syntactic) operations for the monad;
2. reduction rules, involving some specified syntactic operations.

Use of a general notion of **signature** managing this dependency.

Outline

Overview

- Syntax = monad L
- Operations = module morphisms $\Sigma(L) \to L$
- 1-signatures specify operations
- 2-signatures specify operations + equations.

Outline

Operations as module morphisms

Application commutes with substitution

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

Categorical formulation

$L \times L$ supports L-substitution	$\rightsquigarrow$	$L \times L$ is a **module over** L
application commutes with substitution	$\rightsquigarrow$	$\mathrm{app} : L \times L \to L$ is a **module morphism**

[Hirschowitz-Maggesi 2007 : Modules over Monads and Linearity]

Examples of modules

module over a monad T: supports the T-monadic substitution

Examples

- T itself
- $M \times N$ for any modules M and N:

$$\forall (t,u) \in M(X) \times N(X), \qquad X \xrightarrow{f} T(Y),$$

$$\boxed{(t,u)[f] = (t[f], u[f])} \in M(Y) \times N(Y)$$

- M'= **derivative** of a module M:

$$M'(X) = M(\overbrace{X \coprod \{\diamond\}}^{X \text{ extended with a fresh variable } \diamond})$$

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

Operations as module morphisms

Case of λ-calculus

Operations = module morphisms = maps commuting with substitution:

Example: λ-calculus

$$\begin{aligned} \mathrm{app} :\quad & L \times L && \to L \\ \mathrm{abs} :\quad & L' && \to L \end{aligned} \qquad \begin{cases} \mathrm{abs}_X : L(X \coprod \{\diamond\}) \to L(X) \\ \hfill t \mapsto \lambda \diamond .t \end{cases}$$

Combine operations into a single one:

$$\boxed{[\mathrm{app}, \mathrm{abs}] : (L \times L) \coprod L' \to L}$$

1-signatures and their models

Definition

A **1-signature** Σ is a (functorial) assignment

$$\boxed{T \mapsto \Sigma(T)} \qquad \text{e.g.} \quad \boxed{\Sigma_{LC}(T) = (T \times T) \coprod T'}$$

monad module over T

Definition

A **model** of a 1-signature Σ is a pair $M = (T, m)$ where

- T is a monad
- $\Sigma(T) \xrightarrow{m} T$ is a module morphism

Example: λ-calculus

$$[\mathrm{app}, \mathrm{abs}] : \Sigma_{LC}(L) \to L$$

Syntax

(suitable notion of model morphism [Hirschowitz-Maggesi 2012]

Definition

The **syntax** specified by a 1-signature Σ is the initial object in its category of models.

Question: Does the syntax exist for every 1-signature?

Answer: No.

Counter-example: $\Sigma(R) = \mathcal{P} \circ R$

Powerset endofunctor on *Set*.

Examples of 1-signatures generating syntax

λ-calculus

Signature	$T \mapsto (T \times T) \times T'$
Model	$(T \times T) \coprod T' \to T$, or $\begin{pmatrix} T \times T \to T \\ T' \to T \end{pmatrix}$
Syntax	initial model: $(L \times L) \coprod L' \xrightarrow[\text{[app,abs]}]{} L$

Language with a constant and a binary operation

Signature	$T \mapsto 1 \coprod (T \times T)$
Model	$1 \coprod (T \times T) \to T$, or $\begin{pmatrix} 1 \to T \\ T \times T \to T \end{pmatrix}$
Syntax	initial model

Can we generalize this pattern?

Initial semantics for algebraic 1-signatures

Definition

Algebraic 1-signatures = 1-signatures built out of derivatives, finite products, disjoint unions, and the 1-signature $\Theta : T \mapsto T$.

Algebraic 1-signatures $\simeq$ binding signatures [Fiore-Plotkin-Turi 1999]
$\Rightarrow$ specification of n-ary operations, possibly binding variables.

Theorem (Hirschowitz-Maggesi 2007)

Syntax exists for any algebraic 1-signature.

Question: Can we enforce some equations in the syntax?

e.g. *commutativity* or *associativity* of a binary operation.

Quotient of algebraic signatures

Theorem (Ahrens-Lafont-Hirschowitz-Maggesi 2018)

Syntax exists for any "quotient" of algebraic 1-signatures.

Example

a *commutative* binary operation $+$:

$$\forall a, b, \quad a + b = b + a$$


What about an **associative** operation?

Outline

Example: a commutative binary operation

Specification of a binary operation

1-signature	$T \mapsto T \times T$
model	$T \times T \xrightarrow{+} T$

Question: What is an appropriate notion of model for a **commutative** binary operation?

Answer: A monad T equipped with a binary operation $T \times T \xrightarrow{+} T$ which is commutative, i.e.:

$$\begin{tikzcd} T \times T \arrow[rr, bend left, "+"] \arrow[r, "\mathrm{swap}"'] & T \times T \arrow[r, "+"'] & T \end{tikzcd} \qquad \text{where} \quad \mathrm{swap}(t, u) = \mathrm{swap}(u, t)$$

Equations

Σ = 1-signature (e.g. binary operation $\Sigma(T) = T \times T$)

Definition

A **Σ-equation** $A \overset{u}{\underset{v}{\rightrightarrows}} B$ is a (functorial) assignment

$$\boxed{M \mapsto \left(A(M) \overset{u_M}{\underset{v_M}{\rightrightarrows}} B(M) \right)}$$

model of Σ parallel pair of 'M'-module morphisms

Example (Binary commutative operation)

$$\Sigma(T) = T \times T$$

$$\begin{tikzcd} T \times T \arrow[d, "+"] \\ T \end{tikzcd} \quad \mapsto \quad \begin{tikzcd} T \times T \arrow[rr, bend left, "+"] \arrow[dr, "\text{swap}"'] & & T \\ & T \times T \arrow[ur, "+"'] & \end{tikzcd}$$

2-signatures and their models

Definition

A **2-signature** is a pair (Σ, E) where

- Σ is a 1-signature for monads
- E is a set of Σ-equations

Definition

A **model** of a 2-signature (Σ, E) consists of:

- a model $M = \Sigma(T) \xrightarrow{m} T$ of Σ s.t.

$$\forall\ A \overset{u}{\underset{v}{\rightrightarrows}} B \ \in E, \quad \boxed{u_M = v_M} : A(M) \to B(M)$$

morphism of models = morphisms between underlying models of Σ.

Initial semantics for algebraic 2-signatures

Theorem (Ahrens-Lafont-Hirschowitz-Maggesi 2019)

*Any **algebraic** 2-signature has an initial model.*

Definition

A 2-signature (Σ, E) is **algebraic** if:

- Σ is algebraic
- E consists of **elementary** Σ-equations

Main examples of elementary Σ-equations

$A \rightrightarrows B$ s.t.

$$A\left(\begin{array}{c}\Sigma(T)\\ \downarrow\\ T\end{array}\right) = \Phi(T) \qquad B\left(\begin{array}{c}\Sigma(T)\\ \downarrow\\ T\end{array}\right) = T$$

for some *algebraic* 1-signature Φ.

Example: fixpoint operator

Definition [AHLM CSL 2018]

A **fixpoint operator** in a monad R is a module morphism $\mathrm{fix} \colon R' \to R$ s.t. for any term $t \in R(X \coprod \{\diamond\})$, $\mathrm{fix}(t) = t[\diamond \mapsto \mathrm{fix}(t)]$

Intuition:

$\mathtt{fix}(t) \quad := \quad$ `let rec ◇ = t in t`

Algebraic 2-signature $(\Sigma_{\mathrm{fix}}, E_{\mathrm{fix}})$ of a fixpoint operator:

$$\Sigma_{\mathrm{fix}}(R) := R' \qquad\qquad E_{\mathtt{fix}} = \left\{ \begin{tikzcd} \begin{array}{c} R' \\ t \end{array} \arrow[r, bend left, "\mathtt{fix}(t)"] \arrow[r, bend right, "{t[\diamond \mapsto \mathtt{fix}(t)]}"'] & R \end{tikzcd} \right\}$$

Outline

Specifying reduction monads

λ-calculus with β-reduction as a reduction monad:

$$\begin{tikzcd} R \arrow[r, shift left, "\sigma"] \arrow[r, shift right, "\tau"'] & L \end{tikzcd}$$

module over L

module morphisms

monad

- vertices $=$ L $=$ initial model of the signature of λ-calculus.
- arrows $=$ R, σ, τ $=$?
  - **Idea**: defined inductively through reduction rules.

$$(\lambda x.t)\, u \to t[x := u] \qquad \frac{t \to t'}{t\, u \to t'\, u} \qquad \ldots$$

Analysis of a reduction rule

Example: binary congruence for application.

metavariables: as a L-module L^4

$$\overbrace{t, t', u, u'} \mapsto \boxed{\dfrac{t \to t' \qquad u \to u'}{t\, u \to t'\, u'}}$$

hypotheses

conclusion

Hypothesis/conclusion = pair of λ-terms using metavariables

- as parallel module morphisms $L^4 \rightrightarrows L$

$$\text{e.g.} \quad t\, u \to t'\, u' : \qquad \begin{array}{l} (t, t', u, u') \mapsto t\, u \\ (t, t', u, u') \mapsto t'\, u' \end{array}$$

- **Generalization**: $L \rightsquigarrow$ any model T of Σ_{LC}, with application denoted by $\mathrm{app} : T \times T \to T$,

$$\text{e.g.} \quad t\, u \to t'\, u' : \qquad \begin{array}{l} (t, t', u, u') \mapsto \mathrm{app}(t, u) \\ (t, t', u, u') \mapsto \mathrm{app}(t', u') \end{array}$$

Reduction rules

Definition

Let Σ = signature for monads (e.g. $\Theta \times \Theta$ for congruence for application).

Definition of Σ-reduction rules

A Σ-**reduction rule** $(\vec{\sigma}, \vec{\tau})$

$$\boxed{\frac{\sigma_1 \to \tau_1 \qquad \ldots \qquad \sigma_n \to \tau_n}{\sigma_0 \to \tau_0}}$$

assigns (functorially) to each Σ-model T:

- $V(T)$ = T-module of metavariables (e.g. $V(T) = T^4$)
- parallel T-module morphisms $V(T) \underset{\tau_{i,T}}{\overset{\sigma_{i,T}}{\rightrightarrows}} T^{\prime\cdots\prime}$

We write

$$\sigma_i, \tau_i : V \to \Theta^{(n_i)} \qquad n_i = \text{number of derivatives}$$

Reduction signatures

Definition

A **reduction signature** is a pair $(\Sigma, \mathfrak{R})$ where

- Σ is a signature for monads
- $\mathfrak{R}$ is a family of Σ-reduction rules

Example: λ-calculus with β-reduction

- $\Sigma = \Theta \times \Theta + \Theta'$ for app and abs.
- Σ-reduction rules:
  - congruence for application
  - congruence for abstraction:

$$\frac{u \to u'}{\lambda x.u \to \lambda x.u'} \rightsquigarrow \frac{\pi_1 \to \pi_2}{\mathsf{abs} \circ \pi_1 \to \mathsf{abs} \circ \pi_2} \qquad T' \times T' \overset{\pi_{1,T}}{\underset{\pi_{2,T}}{\rightrightarrows}} T'$$

  - β-reduction

Models

Definition

A **model** of a signature $(\Sigma, \mathfrak{R})$ consists of:

- a reduction monad $R \overset{\sigma}{\underset{\tau}{\rightrightarrows}} T$ with a Σ-model structure on T
- for each reduction rule

$$\boxed{\dfrac{\sigma_1 \to \tau_1 \qquad \dots \qquad \sigma_n \to \tau_n}{\sigma_0 \to \tau_0}} \qquad V \overset{\sigma_i}{\underset{\tau_i}{\rightrightarrows}} \Theta^{(n_i)} \qquad \text{in } \mathfrak{R},$$

  - a mapping, for each $v \in V(T)(X)$,

$$\begin{pmatrix} \sigma_1(v) \xrightarrow{r_1} \tau_1(v) \\ \dots \\ \sigma_n(v) \xrightarrow{r_n} \tau_n(v) \end{pmatrix} \quad \mapsto \quad \sigma_0(v) \xrightarrow{op(r_1,\dots r_n)} \tau_0(v)$$

  - compatible with substitution:

$$op(r_1, \dots r_n)[f] = op(r_1[f], \dots, r_n[f])$$

Initiality

(appropriate notion of model morphisms)

Theorem

Σ has an initial model (e.g. Σ is algebraic) $\Rightarrow (\Sigma, \mathfrak{R})$ has an initial model.

Examples

- The reduction signature of the previous slide for λ-calculus with β-reduction has an initial model.
- λ-calculus with explicit substitution [Kesner 2009].

A Theory of Explicit Substitutions with Safe and Full Composition

Generalizing from graphs to bipartite graphs yields more examples:

Examples

- (big step) cbv λ-calculus.
- π-calculus

Summary

- The first main message of your talk in one or two lines.
- The second main message of your talk in one or two lines.
- Perhaps a third message, but not more than that.

- Outlook
  - What we have not done yet.
  - Even more stuff.

For Further Reading I

A. Author.
Handbook of Everything.
Some Press, 1990.

S. Someone.
On this and that.
Journal on This and That. 2(1):50–100, 2000.