Specifying syntax with auxiliary operations

Tom Hirschowitz and Ambroise Lafont

October 13, 2022

Motivation

Mathematical operational semantics (Turi and Plotkin, 1997)

Methods in operational semantics $\longrightarrow$ categorical theorems
or
How about doing operational semantics categorically?

Motivation

In Borthelle et al. (2020):

- Abstract congruence result for applicative bisimilarity in higher-order languages.
- Based on Howe's method.

Bisimulation candidate

- Defined by induction on basic λ-calculus syntax.
- Basic property: closed under substitution.

$\rightsquigarrow$ Need to abstractly specify:

- basic operations in the syntax and
- auxiliary operations on it.

How to specify both kinds of operations, abstractly?

- Solution: Fiore, Plotkin, and Turi (FPT), 1999.
- However: scope limited to capture-avoiding substitution.
- Goal: cover languages with, e.g., context application (e.g., delimited continuations, higher-order process calculi).

This work

Generalise FPT to auxiliary operations other than substitution.
Examples:

- context application: $E[e]$,
- named substitution in $\lambda\mu$: $(e)_\alpha f$ (Vaux's notation),
- partial differentiation in λ-diff: $\frac{\partial e}{\partial x}$, and so on.

Contributions

- Abstract notion of syntax with auxiliary operations: admissible monad morphism.
- Admissible morphisms from distributive laws.
- Toolkit for generating admissible morphisms:
  - simple and incremental structural laws, and
  - structural equational systems.

Plan

1. Introduction
2. Initial-algebra semantics
3. Admissible monad morphisms
4. Simple structural laws
5. Conclusion

Initial-algebra semantics

- Idea: identify syntactic object up to isomorphism, as initial object in a category of models.
- Not tied to any construction method.
- Initiality provides recursion principle.
- But needs initial object to exist.
  - Typical case $\Sigma\text{-}\mathbf{alg} \to \mathbf{C}$ for endofunctor Σ on $\mathbf{C}$.
  - More general: $T\text{-}\mathbf{Alg} \to \mathbf{C}$ for monad T on $\mathbf{C}$.
    Initial algebra: $T\varnothing$.

(Under suitable hypotheses...)

For us:

Syntax, language,... = monad.

Admissible monad morphisms

Definition

A monad morphism $\alpha : R \to S$ is admissible iff $\alpha_\varnothing : R\varnothing \to S\varnothing$ is an isomorphism.

Otherwise said

$R\varnothing$ is an initial S-algebra,

i.e., S-operations are admissible in $R\varnothing$.

Prime class of examples

Fiore, Plotkin, and Turi (FPT), 1999

- Monoidal category $\mathbf{C}$.
- Pointed strong endofunctor Σ.

$\rightsquigarrow$

- Free monad $(I + \Sigma)^*$ for basic syntax.
- Monad $\Sigma^{\circledast}$ for whole syntax.

Theorem (FPT, 1999)

The monad morphism $(I + \Sigma)^* \to \Sigma^{\circledast}$ is admissible.

Distributive laws

Well-known:

Proposition

Any distributive law $\delta : T \circ S \to S \circ T$ induces monad structure on $S \circ T$, with multiplication

$$STST \xrightarrow{S\delta T} SSTT \xrightarrow{\mu^S \mu^T} ST.$$

Furthermore, $S\eta^T : S \to ST$ is a monad morphism.

Distributive laws

Easy:

Proposition

For any distributive law $\delta : T \circ S \to S \circ T$, if $T\varnothing \cong \varnothing$, then $S\eta^T : S \to ST$ is admissible.

Proof.

The morphism $\eta^T_\varnothing : \varnothing \to T\varnothing$ is an iso. □

Intuition for $T\varnothing \cong \varnothing$: no constants!
Example:

- T non-empty multiset monad.
- $T\varnothing$ is non-empty multisets of... nothing.

Admissible morphisms in terms of forgetful functors

- Recall: monadic functor $\mathbf{E} \to \mathbf{C}$ means $\mathbf{E} \cong T\text{-}\mathbf{Alg}$ for some T.
- Let $\mathbf{Monadic}/\mathbf{C}$ denote the full subcategory of $\mathbf{CAT}/\mathbf{C}$ spanned by monadic functors.
- "Semantics" functor

$$\begin{array}{rcl} \mathbf{sem}\colon \mathbf{Mnd}(\mathbf{C})^{op} & \to & \mathbf{Monadic}/\mathbf{C} \\ T & \mapsto & T\text{-}\mathbf{Alg} \\ (\alpha\colon S \to T) & \mapsto & \begin{array}{rcl} T\text{-}\mathbf{Alg} & \to & S\text{-}\mathbf{Alg} \\ (TX \xrightarrow{a} X) & \mapsto & (SX \xrightarrow{\alpha_X} TX \xrightarrow{a} X). \end{array} \end{array}$$

Lemma (Barr? Street?)

The functor $\mathbf{sem}$ *is an equivalence of categories.*

Admissible morphisms in terms of forgetful functors

Proposition

Given a category $\mathbf{C}$ with initial object, monads S and T on $\mathbf{C}$, and a monad morphism $\alpha\colon S \to T$, the following are equivalent:

1. α is admissible;
2. $\mathbf{sem}(\alpha)\colon T\text{-}\mathbf{Alg} \to S\text{-}\mathbf{Alg}$ preserves the initial object.

Simple structural laws: Peano

- Consider $\Sigma(X) = 1 + X$, and let $S = \Sigma^*$.
- $S\varnothing =$ Peano integers.
- Auxiliary operation: addition, defined recursively by

$$0 + e = e \qquad\qquad s(e_1) + e_2 = s(e_1 + e_2). \tag{1}$$

- Refine arity of addition into bifunctor $\Gamma(X, Y) = X \times Y$ for distinguishing the decreasing occurrence.
- View recursive equations as a natural transformation:

$$\begin{array}{rcl} \delta_{X,Y} : \Gamma(\Sigma(X), Y) & \to & S(\Gamma(X, SY) + X + Y) \\ 0 + y & \mapsto & y \\ s(x) + y & \mapsto & s(x + y) \end{array}$$

or in full detail:

$$\begin{array}{rcl} (in_1(\star), y) & \mapsto & \eta^S(in_3(y)) \\ (in_2(x), y) & \mapsto & s(\eta^S(in_1(x, \eta^S(y)))) \end{array}$$

.

Simple structural laws

Let $\Sigma\colon \mathbf{C} \to \mathbf{C}$ and $\Gamma\colon \mathbf{C}^2 \to \mathbf{C}$, with $S = \Sigma^*$.

Definition

Simple structural law: natural transformation

$$\delta_{X,Y} : \Gamma(\Sigma(X), Y) \to S(\Gamma(X, S(Y)) + X + Y).$$

Algebras of a simple structural law

Definition

A δ-algebra: object X with $a\colon \Sigma(X) \to X$ and $b\colon \Gamma(X,X) \to X$ "satisfying the recursive equation":

$$\begin{tikzcd}
\Gamma(\Sigma(X),X) \arrow[rrr, "\Gamma(a,X)"] \arrow[d, "\delta_{X,Y}"'] & & & \Gamma(X,X) \arrow[d, "b"] \\
S(\Gamma(X,S(X))+X+X) \arrow[r, "S(\Gamma(X,\bar{a})+X+X)"'] & S(\Gamma(X,X)+X+X) \arrow[r, "S[b,X,X]"'] & S(X) \arrow[r, "\bar{a}"'] & X
\end{tikzcd}$$

δ-**alg** $=$ category of δ-algebras.

Forgetful functors:

$$\begin{tikzcd}
\Sigma\text{-}\mathbf{alg} \arrow[dr] & & \delta\text{-}\mathbf{alg} \arrow[ll] \arrow[dl] \\
& \mathbf{C} &
\end{tikzcd}$$

Admissible morphisms from simple structural laws

Theorem

For any $\delta_{X,Y} : \Gamma(\Sigma(X), Y) \to S(\Gamma(X, S(Y)) + X + Y)$

with $\mathbf{C}$ *locally finitely presentable,* Σ *finitary, and* Γ *cocontinuous in its first argument and finitary in its second one,*

$$\begin{tikzcd}
\Sigma\text{-}\mathbf{alg} \arrow[dr] & & \delta\text{-}\mathbf{alg} \arrow[ll, hook', "\text{preserves initial object}"'] \arrow[dl, "\text{finitary and monadic}"] \\
& \mathbf{C}. &
\end{tikzcd}$$

Corollary

The induced monad morphism $\Sigma^* \to \delta^*$ *is admissible.*

Application: sharing λ-calculus (Accattoli, Kesner,...)

Syntax:

$$\begin{array}{rcl} e, f & ::= & x \mid e\ f \mid \lambda x.e \mid e\langle x \mapsto f\rangle \\ E & ::= & \square \mid E\langle x \mapsto f\rangle \end{array}$$

Context application:

$$\begin{array}{rcl} \square[e] & = & e \\ (E\langle x \mapsto f\rangle)[e] & = & E[e]\langle x \mapsto f\rangle \end{array}$$

Warning, variable capture

$(\square\langle x \mapsto f\rangle)[x] = x\langle x \mapsto f\rangle$.

Base category

Functor category $[\mathbb{F}, \mathbf{Set}^{1+\mathbb{N}}]$, where $\mathbb{F}$ = finite ordinals and all maps.

Notation

- $\mathbf{p} = in_1(\star) \in 1 + \mathbb{N}$ (program).
- $\mathbf{c}_m = in_2(m) \in 1 + \mathbb{N}$ (context with m capturing variables).
- For any $X \in [\mathbb{F}, \mathbf{Set}^{1+\mathbb{N}}]$, $n \in \mathbb{F}$, $m \in \mathbb{N}$:
  - $X(n)_{\mathbf{p}}$: programs with n free variables, and
  - $X(n)_{\mathbf{c}_m}$: contexts with n free variables and m capturing variables.

Syntax (with notation)

Programs

$$\Sigma(X)(n)_{\mathbf{p}} = n + X(n)^2_{\mathbf{p}} + X(n+1)_{\mathbf{p}} + X(n+1)_{\mathbf{p}} \times X(n)_{\mathbf{p}}$$

$$e ::= x \mid e\ e \mid \lambda(e) \mid e\langle e\rangle$$

Contexts

$$\Sigma(X)(n)_{\mathbf{c}_{1+m}} = X(n+1)_{\mathbf{c}_m} \times X(n)_{\mathbf{p}}$$

$$\Sigma(X)(n)_{\mathbf{c}_0} = 1$$

$$E ::= E\langle x \mapsto e\rangle \mid \square$$

Syntax (with notation)

Contexts

$$\begin{aligned} \Sigma(X)(n)_{\mathbf{c}_{1+m}} &= X(n+1)_{\mathbf{c}_m} \times X(n)_{\mathbf{p}} \\ \Sigma(X)(n)_{\mathbf{c}_0} &= 1 \end{aligned} \qquad \begin{aligned} E &::= E\langle x \mapsto e\rangle \\ &\mid \square \end{aligned}$$

Remark: typing $E\langle x \mapsto e\rangle$.

- Σ-algebra structure on X yields

$$\begin{aligned} X(n+1)_{\mathbf{c}_m} \times X(n)_{\mathbf{p}} &\to X(n)_{\mathbf{c}_{1+m}} \\ (E, f) &\mapsto E\langle x \mapsto f\rangle \end{aligned}$$

- x is free in E, but capturing in $E\langle x \mapsto f\rangle$.
- Think of

$$\begin{array}{rcrcrcr} n & ; & m & \text{as} & 1,\ldots,n & ; & n+1,\ldots,n+m \\ n+1 & ; & m & \text{as} & 1,\ldots,n+1 & ; & n+1+1,\ldots,n+1+m \\ n & ; & m+1 & \text{as} & 1,\ldots,n & ; & n+1,n+2,\ldots,n+(m+1) \end{array}$$

Simple structural law

Take
$$\begin{aligned} \Gamma(X,Y)(n)_{\mathbf{c}_m} &= \varnothing \\ \Gamma(X,Y)(n)_{\mathbf{p}} &= \textstyle\sum_{m\in\mathbb{N}} X(n)_{\mathbf{c}_m} \times Y(n+m)_{\mathbf{p}} \end{aligned}$$
and

$$\begin{aligned} \Sigma(X)(n)_{\mathbf{c}_m} \times Y(n+m)_{\mathbf{p}} &\longrightarrow S(\Gamma(X,S(Y)) + X + Y)(n)_{\mathbf{p}} \\ (\square, e) &\mapsto e && (\text{if } m = 0) \\ (E\langle f\rangle, e) &\mapsto E[e]\ \langle f\rangle && (\text{if } m = m'+1), \end{aligned}$$

for all $n, m \in \mathbb{N}$.

Remark

Both $\underbrace{E}_{(n+1,m')}\ [\underbrace{e}_{n+1+m'}]$ *and* $\underbrace{E\langle f\rangle}_{(n,1+m')}\ [\underbrace{e}_{n+1+m'}]$ *are valid.*

Characterisation of induced monad

Let $\delta_{X,Y} : \Gamma(\Sigma(X), Y) \to S(\Gamma(X, S(Y)) + X + Y)$ as in the theorem.

Theorem

The monad corresponding to δ-$\mathbf{alg} \to \mathbf{C}$ is induced by a distributive law

$$TS \to ST,$$

with $T = (X \mapsto \Gamma(X, SX))^*$

Intuition: auxiliary operations, with arbitrary S-terms in non-decreasing occurrences.

Conclusion

In this talk:

- Admissible monad morphisms.
- Distributive laws $\longrightarrow$ admissible monad morphisms.
- Simple structural laws $\longrightarrow$ admissible monad morphisms.
- Application: context application in sharing λ-calculus.

Not in this talk:

- Incremental structural laws $\longrightarrow$ admissible monad morphisms.
- Structural equational systems: benign, admissible quotienting.
- Applications:
  - named substitution in $\lambda\mu$,
  - partial differentiation in λ-diff,
  - Fiore, Plotkin, and Turi.

Nowhere yet: quotienting basic syntax, compatibly with auxiliary operations.

Indices in $E\langle f\rangle$

- Let $E_1 \quad = \quad (x_1, x_2; x_3 \vdash \Box\langle x_3 \mapsto f\rangle : \mathbf{c})$.
- Let $E = E_1\langle x_2 \mapsto g\rangle \quad = \quad (x_1; x_2, x_3 \vdash \Box\langle x_3 \mapsto f\rangle\langle x_2 \mapsto g\rangle : \mathbf{c})$.
- Let $e \quad = \quad (x_1, x_2, x_3 \vdash x_1\ x_2\ x_3 : \mathbf{p})$.

Form

- $x_1, x_2 \vdash E_1[e] = (x_1\ x_2\ x_3)\langle x_3 \mapsto f\rangle : \mathbf{p}$
- $x_1 \vdash E[e] = (x_1\ x_2\ x_3)\langle x_3 \mapsto f\rangle\langle x_2 \mapsto g\rangle : \mathbf{p}$.