

# Algebraic operations and $\lambda$ -calculus

---

Renato Neves



Universidade do Minho



# Table of Contents

Background

Integration of algebraic operations in  $\lambda$ -calculus

Semantics of  $\lambda$ -calculus with algebraic operations

## Recalling $\lambda$ -Calculus

$$\mathbb{A} \ni 1 \mid \mathbb{A} \times \mathbb{A} \mid \mathbb{A} \rightarrow \mathbb{A}$$

$$\frac{x : \mathbb{A} \in \Gamma}{\Gamma \vdash x : \mathbb{A}}$$

$$\frac{}{\Gamma \vdash * : 1}$$

$$\frac{\Gamma \vdash V : \mathbb{A} \times \mathbb{B}}{\Gamma \vdash \pi_1 V : \mathbb{A}}$$

$$\frac{\Gamma \vdash V : \mathbb{A} \quad \Gamma \vdash U : \mathbb{B}}{\Gamma \vdash \langle V, U \rangle : \mathbb{A} \times \mathbb{B}}$$

$$\frac{\Gamma, x : \mathbb{A} \vdash V : \mathbb{B}}{\Gamma \vdash \lambda x : \mathbb{A}. V : \mathbb{A} \rightarrow \mathbb{B}}$$

$$\frac{\Gamma \vdash V : \mathbb{A} \rightarrow \mathbb{B} \quad \Gamma \vdash U : \mathbb{A}}{\Gamma \vdash V U : \mathbb{B}}$$

# Sequential Composition

A “new” deductive rule

$$\frac{\Gamma \vdash V : \mathbb{A} \quad x : \mathbb{A} \vdash U : \mathbb{B}}{\Gamma \vdash x \leftarrow V ; U : \mathbb{B}}$$

It reads as “bind the computation  $V$  to  $x$  and then run  $U$ ”

Interpretation defined as

$$\frac{\llbracket \Gamma \vdash V : \mathbb{A} \rrbracket = f \quad \llbracket x : \mathbb{A} \vdash U : \mathbb{B} \rrbracket = g}{\llbracket \Gamma \vdash x \leftarrow V ; U : \mathbb{B} \rrbracket = g \cdot f}$$

# Table of Contents

Background

Integration of algebraic operations in  $\lambda$ -calculus

Semantics of  $\lambda$ -calculus with algebraic operations

# Signatures

## Signature

A set  $\Sigma = \{(\sigma_1, n_1), (\sigma_2, n_2), \dots\}$  of operations  $\sigma_i$  paired with the **number** of inputs  $n_i$  they are supposed to receive

Signatures will later be integrated in  $\lambda$ -calculus

They constitute the aforementioned **algebraic operations**

## Examples

- Exceptions:  $\{(e, 0)\}$
- Read a bit from the environment:  $\{(\text{read}, 2)\}$
- Wait calls:  $\{(\text{wait}_n, 1) \mid n \in \mathbb{N}\}$
- Non-deterministic choice:  $\{(+, 2)\}$

## Algebraic operations in $\lambda$ -calculus

We choose a signature  $\Sigma$  of algebraic operations and introduce a new deductive rule

$$\frac{(\sigma, n) \in \Sigma \quad \forall i \leq n. \Gamma \vdash M_i : \mathbb{A}}{\Gamma \vdash \sigma(M_1, \dots, M_n) : \mathbb{A}}$$

## Examples of effectful $\lambda$ -terms

- $x : \mathbb{A} \vdash \text{wait}_1(x) : \mathbb{A}$  – adds **delay** of one second to returning  $x$
- $\Gamma \vdash e() : \mathbb{A}$  – raises an **exception**  $e$
- $\Gamma \vdash \text{write}_v(M) : \mathbb{A}$  – writes  $v$  in **memory** and then runs  $M$
- $x : \mathbb{A} \times \mathbb{A} \vdash \text{read}(\pi_1 x, \pi_2 x) : \mathbb{A}$  – **receives** a bit: if the bit is 0 it returns  $\pi_1 x$  otherwise it returns  $\pi_2 x$



## Examples of effectful $\lambda$ -terms

- $x : \mathbb{A} \vdash \text{wait}_1(x) : \mathbb{A}$  – adds **delay** of one second to returning  $x$
- $\Gamma \vdash e() : \mathbb{A}$  – raises an **exception**  $e$
- $\Gamma \vdash \text{write}_v(M) : \mathbb{A}$  – writes  $v$  in **memory** and then runs  $M$
- $x : \mathbb{A} \times \mathbb{A} \vdash \text{read}(\pi_1 x, \pi_2 x) : \mathbb{A}$  – **receives** a bit: if the bit is 0 it returns  $\pi_1 x$  otherwise it returns  $\pi_2 x$

### Exercise

Define a  $\lambda$ -term  $x : \mathbb{A} \vdash ? : \mathbb{A}$  that requests a bit from the user and depending on the value read it returns  $x$  with either one or two seconds of delay.

# Table of Contents

Background

Integration of algebraic operations in  $\lambda$ -calculus

Semantics of  $\lambda$ -calculus with algebraic operations

# Semantics of $\lambda$ -Calculus with algebraic Operations

How to provide semantics to these programming languages?

Short answer: via monads

Long answer: see the next slides ...

# The core idea

Programs  $\Gamma \vdash V : \mathbb{A}$  interpreted as functions

$$\llbracket \Gamma \vdash V : \mathbb{A} \rrbracket : \llbracket \Gamma \rrbracket \longrightarrow \llbracket \mathbb{A} \rrbracket$$

... and there exists **only one** function of type

$$\llbracket \Gamma \rrbracket \longrightarrow \llbracket 1 \rrbracket$$

Problem: it is then necessarily the case that

$$\llbracket \Gamma \vdash x : 1 \rrbracket = \llbracket \Gamma \vdash \text{wait}_1(x) : 1 \rrbracket$$

despite these programs having different execution times

## The core idea pt. II

Interpreted a program  $\Gamma \vdash V : \mathbb{A}$  as a function

$$\llbracket \Gamma \vdash V : \mathbb{A} \rrbracket : \llbracket \Gamma \rrbracket \longrightarrow \llbracket \mathbb{A} \rrbracket$$

which returns values in  $\llbracket \mathbb{A} \rrbracket$ . But values now come with effects ...

Instead of having  $\llbracket \mathbb{A} \rrbracket$  as set of outputs, we will have a set  $T\llbracket \mathbb{A} \rrbracket$  of **effectful values**

$$\llbracket \Gamma \vdash M : \mathbb{A} \rrbracket : \llbracket \Gamma \rrbracket \longrightarrow T\llbracket \mathbb{A} \rrbracket$$

$T$  should thus be a **set-constructor**: given a set of outputs  $X$  it returns a set of effectful values  $TX$  over  $X$

## The core idea pt. III

For wait calls, the corresponding set-constructor  $T$  is defined as

$$X \mapsto \mathbb{N} \times X$$

*i.e.* values in  $X$  paired with an **execution time**

For exceptions, the corresponding set-constructor  $T$  is defined as

$$X \mapsto X + \{e\}$$

*i.e.* values in  $X$  plus an element  $e$  **representing the exception**

## Another problem

This idea of a set-constructor  $T$  seems good, but it breaks sequential composition

$$\begin{aligned} \llbracket \Gamma \vdash M : A \rrbracket & : \llbracket \Gamma \rrbracket \rightarrow T[A] \\ \llbracket x : A \vdash N : B \rrbracket & : [A] \rightarrow T[B] \end{aligned}$$

We need a way to convert a function  $h : X \rightarrow TY$  into a function of the type

$$h^* : TX \rightarrow TY$$

## Another problem pt. II

There are set-constructors  $T$  for which this is possible

In the case of **wait-calls**

$$\frac{f : X \rightarrow TY = \mathbb{N} \times Y}{f^*(n, x) = (n + m, y) \text{ where } f(x) = (m, y)}$$

In the case of **exceptions**

$$\frac{f : X \rightarrow TY = Y + \{e\}}{f^*(x) = f(y) \quad f^*(e) = e}$$



$$\begin{aligned} & \llbracket x : 1 \vdash y \leftarrow \text{wait}_1(x); \text{wait}_2(y) : 1 \rrbracket \\ &= \llbracket y : 1 \vdash \text{wait}_2(y) : 1 \rrbracket^* \cdot \llbracket x : 1 \vdash \text{wait}_1(x) : 1 \rrbracket \\ &= (v \mapsto (2, v))^* \cdot (v \mapsto (1, v)) \\ &= v \mapsto (3, v) \end{aligned}$$

## Yet another problem

Idea of interpreting  $\lambda$ -terms  $\Gamma \vdash M : \mathbb{A}$  as functions

$$\llbracket \Gamma \vdash M : \mathbb{A} \rrbracket : \llbracket \Gamma \rrbracket \longrightarrow T[\mathbb{A}]$$

looks good but it presupposes that all terms invoke effects

Some terms do not do this, e.g.

$$\llbracket x : \mathbb{A} \vdash x : \mathbb{A} \rrbracket : \llbracket \mathbb{A} \rrbracket \longrightarrow \llbracket \mathbb{A} \rrbracket$$

### Solution

$T[\mathbb{A}]$  should include **effect-free** values, and we should have

$$\eta_{[\mathbb{A}]} : \llbracket \mathbb{A} \rrbracket \longrightarrow T[\mathbb{A}]$$

which maps a value to its effect-free representation

## Yet another problem pt. II

Again there are set-constructors  $T$  for which this is possible:

In the case of **wait-calls**

$$\frac{TX = \mathbb{N} \times X}{\eta_X(x) = (0, x)}$$

(i.e. no wait call was invoked)

In the case of **exceptions**

$$\frac{TX = X + \{e\}}{\eta_X(x) = x}$$

(i.e. the exception  $e$  was never raised)

# Monads unlocked!!

Our previous analysis **naturally** leads to the notion of a **monad**

## Monad

A triple  $(T, \eta, (-)^*)$  where  $T$  is a set-constructor,  $\eta$  a function  $\eta_X : X \rightarrow TX$  for each set  $X$ , and  $(-)^*$  an operation

$$\frac{f : X \rightarrow TY}{f^* : TX \rightarrow TY}$$

s.t. the following laws hold:  $\eta^* = \text{id}$ ,  $f^* \cdot \eta = f$ ,  $(f^* \cdot g)^* = f^* \cdot g^*$

These laws are required to forbid “weird” computational behaviour

## Exercise

Show that the set-constructor

$$X \mapsto \mathbb{N} \times X$$

can be equipped with a monadic structure

Show that the set-constructor

$$X \mapsto X + 1$$

can be equipped with a monadic structure

## To Keep In Mind

Let us use what we learned to extend  $\lambda$ -calculus with algebraic operations and provide it with a proper semantics

Recall that,

- we fix a signature  $\Sigma$  of algebraic operations
- we have monads  $(T, \eta, (-)^*)$  at our disposal
- Programs  $\Gamma \vdash V : \mathbb{A}$  can be seen either as functions of type  $[[\Gamma]] \rightarrow [[\mathbb{A}]]$  or of type  $[[\Gamma]] \rightarrow T[[\mathbb{A}]]$

# Semantics for Effectful Simply-Typed $\lambda$ -Calculus

Types  $\mathbb{A}$  are interpreted as **sets**  $\llbracket \mathbb{A} \rrbracket$

$$\llbracket 1 \rrbracket = \{\star\} \quad \llbracket \mathbb{A} \times \mathbb{B} \rrbracket = \llbracket \mathbb{A} \rrbracket \times \llbracket \mathbb{B} \rrbracket \quad \llbracket \mathbb{A} \rightarrow \mathbb{B} \rrbracket = (T\llbracket \mathbb{B} \rrbracket)^{\llbracket \mathbb{A} \rrbracket}$$

A typing context  $\Gamma$  is interpreted as

$$\llbracket \Gamma \rrbracket = \llbracket x_1 : \mathbb{A}_1, \dots, x_n : \mathbb{A}_n \rrbracket = \llbracket \mathbb{A}_1 \rrbracket \times \dots \times \llbracket \mathbb{A}_n \rrbracket$$

For each operation  $(\sigma, n) \in \Sigma$  and set  $X$  we postulate the existence of a map

$$\llbracket \sigma \rrbracket_X : (TX)^n \longrightarrow TX$$

# Semantics for effectful simply-typed $\lambda$ -calculus II

$$\frac{x_i : \mathbb{A} \in \Gamma}{\llbracket \Gamma \vdash x_i \rrbracket = \pi_i}$$

$$\frac{}{\llbracket \Gamma \vdash * \rrbracket = !}$$

$$\frac{\llbracket \Gamma \vdash V : \mathbb{A} \rrbracket = f \quad \llbracket \Gamma \vdash U : \mathbb{B} \rrbracket = g}{\llbracket \Gamma \vdash \langle V, U \rangle : \mathbb{A} \times \mathbb{B} \rrbracket = \langle f, g \rangle}$$

$$\frac{\llbracket \Gamma, x : \mathbb{A} \vdash_c M : \mathbb{B} \rrbracket = f}{\llbracket \Gamma \vdash \lambda x : \mathbb{A}. M : \mathbb{A} \rightarrow \mathbb{B} \rrbracket = \lambda f}$$

$$\frac{\llbracket \Gamma \vdash V : \mathbb{A} \times \mathbb{B} \rrbracket = f}{\llbracket \Gamma \vdash \pi_1 V : \mathbb{A} \rrbracket = \pi_1 \cdot f}$$

$$\frac{\llbracket \Gamma \vdash V : \mathbb{A} \rrbracket = f}{\llbracket \Gamma \vdash_c \text{return } V : \mathbb{A} \rrbracket = \eta \cdot f}$$

$$\frac{\llbracket \Gamma \vdash_c M : \mathbb{A} \rrbracket = f \quad \llbracket x : \mathbb{A} \vdash_c N : \mathbb{B} \rrbracket = g}{\llbracket \Gamma \vdash_c x \leftarrow M ; N : \mathbb{B} \rrbracket = g^* \cdot f}$$

$$\frac{\llbracket \Gamma \vdash V : \mathbb{A} \rightarrow \mathbb{B} \rrbracket = f \quad \llbracket \Gamma \vdash U : \mathbb{A} \rrbracket = g}{\llbracket \Gamma \vdash_c V U : \mathbb{B} \rrbracket = \text{app} \cdot \langle f, g \rangle}$$

$$\frac{(\sigma, n) \in \Sigma \quad \forall i \leq n. \llbracket \Gamma \vdash_c M_i : \mathbb{A} \rrbracket = f_i}{\llbracket \Gamma \vdash_c \sigma(M_1, \dots, M_n) \rrbracket = \llbracket \sigma \rrbracket_{[\mathbb{A}]} \cdot \langle f_1, \dots, f_n \rangle}$$