Programming with algebraic effects

Renato Neves

Architecture and Calculi Course Unit

Table of Contents

Overview

Modern programming typically involves different effects

- memory cell manipulation
- read/print calls
- exception raising operations
- probabilistic operations
- wait calls
- interaction with physical processes

In the following lectures we will study the mathematical foundations of

$$\boxed{\text{Effectful Programming}}$$

in a uniform way

Table of Contents

The essentials of programming

In order to study effectful programming, we should think of what are the basic features of (higher-order) programming ...

- variables
- function application
- function abstraction
- pairing ...

and base our study on the simplest programming language containing these features ...

$$\boxed{\text{Simply-typed } \lambda\text{-calculus}}$$

It is the basis of HASKELL, ML, EFF, F#, AGDA, ELM and many other programming languages.

Simply-typed λ-calculus

Types:

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

Programs are built according to the inference rules:

$$\frac{x : \mathbb{A} \in \Gamma}{\Gamma \vdash x : \mathbb{A}} \ (\mathrm{var}) \qquad \frac{}{\Gamma \vdash * : 1} \ (\mathrm{unit}) \qquad \frac{\Gamma \vdash V : \mathbb{A} \times \mathbb{B}}{\Gamma \vdash \pi_1 V : \mathbb{A}} \ (\mathrm{prj})$$

$$\frac{\Gamma \vdash V : \mathbb{A} \qquad \Gamma \vdash U : \mathbb{B}}{\Gamma \vdash \langle V, U \rangle : \mathbb{A} \times \mathbb{B}} \ (\mathrm{prod}) \qquad \frac{\Gamma, x : \mathbb{A} \vdash V : \mathbb{B}}{\Gamma \vdash \lambda x : \mathbb{A}.\, V : \mathbb{A} \to \mathbb{B}} \ (\mathrm{abs})$$

$$\frac{\Gamma \vdash V : \mathbb{A} \to \mathbb{B} \quad \Gamma \vdash U : \mathbb{A}}{\Gamma \vdash V\, U : \mathbb{B}} \ (\mathrm{app})$$

Γ is a non-repetitive list of typed variables $x_1 : \mathbb{A}_1 \ldots x_n : \mathbb{A}_n$.

Examples of λ-terms

$\lambda x : \mathbb{A}.\, x : \mathbb{A} \to \mathbb{A}$ (identity)

$\lambda x : \mathbb{A}.\, \langle x, x \rangle : \mathbb{A} \to \mathbb{A} \times \mathbb{A}$ (duplication)

$\lambda V : \mathbb{A} \times \mathbb{B}.\, \langle \pi_2 V, \pi_1 V \rangle : \mathbb{A} \times \mathbb{B} \to \mathbb{B} \times \mathbb{A}$ (swap)

$\lambda f : \mathbb{A} \to \mathbb{B}, \lambda g : \mathbb{B} \to \mathbb{C}, \lambda x : \mathbb{A}.\, g(f\ x) : \ldots$ (composition)

Exercise

Build a λ-term (using the inference rules) that takes a variable $f : \mathbb{A} \to \mathbb{A}$, a variable $x : \mathbb{A}$, and applies f to x twice.

Semantics for simply-typed λ-calculus

We wish to assign a mathematical meaning to λ-terms

$$[\![-]\!] \colon \lambda\text{-Terms} \longrightarrow \ldots$$

so that we can reason about them in a rigorous way, and take advantage of known mathematical theories

Semantics for simply-typed λ-calculus

We wish to assign a mathematical meaning to λ-terms

$$[\![-]\!]\colon \lambda\text{-Terms} \longrightarrow \ldots$$

so that we can reason about them in a rigorous way, and take advantage of known mathematical theories

This is the goal of the next slides: we will study how to interpret λ-terms as functions. But first . . .

Basic facts about functions

For every set X, there is a "trivial" function

$$! : X \longrightarrow \{\star\} = 1, \qquad !\,(x) = \star$$

We can always pair two functions $f : X \to A$, $g : X \to B$ into

$$\langle f, g\rangle : X \to A \times B, \qquad \langle f, g\rangle(x) = (f\,x, g\,x)$$

Consider two sets X, Y. The exist "projection" functions

$$\pi_1 : X \times Y \to X, \qquad \pi_1(x, y) = x$$
$$\pi_2 : X \times Y \to Y, \qquad \pi_2(x, y) = y$$

Basic facts about functions

We can always 'curry' a function $f : X \times Y \to Z$ into

$$\lambda f : X \to Z^Y, \qquad \lambda f(a) = (b \mapsto f(a, b))$$

Consider two sets X, Y. There exists an "application" function

$$\mathrm{app} : Z^Y \times Y \to Z, \qquad \mathrm{app}(f, y) = f\ y$$

Functional semantics for the simply-typed λ-calculus

Types $\mathbb{A}$ are interpreted as sets $[\![\mathbb{A}]\!]$

$$[\![1]\!] = \{\star\}$$
$$[\![\mathbb{A} \times \mathbb{B}]\!] = [\![\mathbb{A}]\!] \times [\![\mathbb{B}]\!]$$
$$[\![\mathbb{A} \to \mathbb{B}]\!] = [\![\mathbb{B}]\!]^{[\![\mathbb{A}]\!]}$$

A typing context Γ is interpreted as

$$[\![\Gamma]\!] = [\![x_1 : \mathbb{A}_1 \times \cdots \times x_n : \mathbb{A}_n]\!] = [\![\mathbb{A}_1]\!] \times \cdots \times [\![\mathbb{A}_n]\!]$$

A λ-term $\Gamma \vdash V : \mathbb{A}$ is interpreted as a function

$$[\![\Gamma \vdash V : \mathbb{A}]\!] : [\![\Gamma]\!] \longrightarrow [\![\mathbb{A}]\!]$$

Functional semantics for the simply-typed λ-calculus

A program term $\Gamma \vdash V : \mathbb{A}$ is interpreted as a function

$$[\![\Gamma \vdash V : \mathbb{A}]\!] : [\![\Gamma]\!] \longrightarrow [\![\mathbb{A}]\!]$$

in the following way

$$\frac{x_i : \mathbb{A} \in \Gamma}{[\![\Gamma \vdash x_i : \mathbb{A}]\!] = \pi_i} \qquad \frac{}{[\![\Gamma \vdash * : 1]\!] = \,!} \qquad \frac{[\![\Gamma \vdash V : \mathbb{A} \times \mathbb{B}]\!] = f}{[\![\Gamma \vdash \pi_1 V : \mathbb{A}]\!] = \pi_1 \cdot f}$$

$$\frac{[\![\Gamma \vdash V : \mathbb{A}]\!] = f \qquad [\![\Gamma \vdash U : \mathbb{B}]\!] = g}{[\![\Gamma \vdash \langle V, U \rangle : \mathbb{A} \times \mathbb{B}]\!] = \langle f, g \rangle} \qquad \frac{[\![\Gamma, x : \mathbb{A} \vdash V : \mathbb{B}]\!] = f}{[\![\Gamma \vdash \lambda x : \mathbb{A}.\, V : \mathbb{A} \to \mathbb{B}]\!] = \lambda f}$$

$$\frac{[\![\Gamma \vdash V : \mathbb{A} \to \mathbb{B}]\!] = f \qquad [\![\Gamma \vdash U : \mathbb{A}]\!] = g}{[\![\Gamma \vdash V\, U : \mathbb{B}]\!] = \mathrm{app} \cdot \langle f, g \rangle}$$

Exercise

Show (using the inference rules) that the equations below hold.

$$
\begin{aligned}
[\![x : \mathbb{A}, y : \mathbb{B} \vdash \pi_1\langle x, y\rangle : \mathbb{A}]\!] &= [\![x : \mathbb{A}, y : \mathbb{B} \vdash x : \mathbb{A}]\!] \\
[\![\Gamma \vdash V : \mathbb{A}]\!] &= [\![\Gamma \vdash \langle \pi_1 V, \pi_2 V\rangle : \mathbb{A}]\!]
\end{aligned}
$$

Time to add algebraic effects to our programming language

Table of Contents

Algebraic theories

An algebraic theory (Σ, E) is a pair where

- $\Sigma = \{\sigma_1 : n_1, \ldots, \sigma_n : m_n\}$ is a set of operations (the effects)
- E is a set of equations that relate the operations

'$\sigma : n$' means that the operation σ receives n arguments

Exceptions

$\Sigma = \{e : 0\},\ \ E = \emptyset$ (no equations)

Read a bit

$\Sigma = \{read : 2\},\ \ E = \emptyset$ (no equations)

Wait calls

$\Sigma = \{\mathrm{wait}_n : 1 \mid n \in \mathbb{N}\}$,
$E = \{\mathrm{wait}_n(\mathrm{wait}_m(x)) = \mathrm{wait}_{n+m}(x) \mid n, m \in \mathbb{N}\}$

Algebraic theories and their algebras

An algebra for an algebraic theory (Σ, E) is a set X equipped with a function $[\![\sigma_i]\!]\colon X^{n_i} \to X$ for each $\sigma_i : n_i$ in Σ such that all equations in E are respected

Exceptions

An algebra for the theory of exceptions is a set X equipped with a function $[\![e]\!]\colon X^0 = 1 \to X$

Read

An algebra for the theory of read calls is a set X equipped with a function $[\![\mathrm{read}]\!]\colon X^2 = X \times X \to X$

Wait calls

...

Table of Contents

Simply-typed λ-calculus with **effects**

Types are defined as before

We choose an equational theory (Σ, E); the operations in Σ correspond to effects

We define a new inference rule

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

Examples of **effectful** λ-terms

$\lambda x : \mathbb{A}.\, \mathrm{wait}_1(x) : \mathbb{A} \to \mathbb{A}$ (waits one second before returning x)

$\lambda x : \mathbb{A}.\, e : \mathbb{A} \to \mathbb{A}$ (raises an exception e)

$\lambda x : \mathbb{A} \times \mathbb{A}.\, \mathrm{read}(\pi_1\, x, \pi_2\, x) : \mathbb{A} \times \mathbb{A} \to \mathbb{A}$ (requests a bit from the user. If the bit is 0 it returns $\pi_1\, x$, otherwise returns $\pi_2\, x$)

Examples of **effectful** λ-terms

$\lambda x : \mathbb{A}.\, \mathrm{wait}_1(x) : \mathbb{A} \to \mathbb{A}$ (waits one second before returning x)

$\lambda x : \mathbb{A}.\, e : \mathbb{A} \to \mathbb{A}$ (raises an exception e)

$\lambda x : \mathbb{A} \times \mathbb{A}.\, \mathrm{read}(\pi_1\, x, \pi_2\, x) : \mathbb{A} \times \mathbb{A} \to \mathbb{A}$ (requests a bit from the user. If the bit is 0 it returns $\pi_1\, x$, otherwise returns $\pi_2\, x$)

Exercise

Define an effectful λ-term $\lambda x : \mathbb{A}.\ \cdots : \mathbb{A} \to \mathbb{A}$ that requests a bit from the user; depending on the value read either waits one or two seconds before returning x

Examples of **effectful** λ-terms

$\lambda x : \mathbb{A}.\, \mathrm{wait}_1(x) : \mathbb{A} \to \mathbb{A}$ (waits one second before returning x)

$\lambda x : \mathbb{A}.\, e : \mathbb{A} \to \mathbb{A}$ (raises an exception e)

$\lambda x : \mathbb{A} \times \mathbb{A}.\, \mathrm{read}(\pi_1\, x, \pi_2\, x) : \mathbb{A} \times \mathbb{A} \to \mathbb{A}$ (requests a bit from the user. If the bit is 0 it returns $\pi_1\, x$, otherwise returns $\pi_2\, x$)

Exercise

Define an effectful λ-term $\lambda x : \mathbb{A}.\ \cdots : \mathbb{A} \to \mathbb{A}$ that requests a bit from the user; depending on the value read either waits one or two seconds before returning x

We could also have considered *e.g.* operations for probabilistic choice and memory cell manipulation

Table of Contents

Semantics for effectful simply-typed λ-calculus

How to provide a suitable semantics to this family of effectful programming languages?

The short answer: via monads

the long answer: see the next slides . . .

The core idea

Previously, we interpreted a term $\Gamma \vdash V : \mathbb{A}$ as a function

$$[\![\Gamma \vdash V : \mathbb{A}]\!] : [\![\Gamma]\!] \longrightarrow [\![\mathbb{A}]\!]$$

which returns values in $[\![\mathbb{A}]\!]$. But now values come with effects...

So instead of having $[\![A]\!]$ as the set of outputs, we have a set of effects $T[\![\mathbb{A}]\!]$ over $[\![\mathbb{A}]\!]$ as outputs

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

T is a 'set-constructor': given a set of values X it returns a set of effects TX over X

The core idea

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

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

The problem

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

$$
\begin{aligned}
[\![\vdash M : \mathbb{A}]\!] &: 1 \to T[\![\mathbb{A}]\!] \\
[\![x : \mathbb{A} \vdash N : \mathbb{B}]\!] &: [\![\mathbb{A}]\!] \to T[\![\mathbb{B}]\!]
\end{aligned}
$$

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

$$h^{\star} : TX \to TY$$

The problem

There are set-constructors T for which this is possible

In the case of exceptions,

$$\frac{f : X \to TY = Y + \{e\}}{f^\star(x) = f(y) \qquad f^\star(e) = e}$$

In the case of wait-calls,

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

The problem

The idea of interpreting λ-terms $\Gamma \vdash M : \mathbb{A}$ as functions

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

looks good but it presupposes that all terms invoke effects

There are terms that do not do this, *e.g.*

$$[\![x : \mathbb{A} \vdash x : \mathbb{A}]\!] \colon [\![\mathbb{A}]\!] \longrightarrow [\![\mathbb{A}]\!]$$

Solution

$T[\![\mathbb{A}]\!]$ should also include values free of effects, and there should exist a function

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

that maps values to the corresponding effect-free representations in $T[\![\mathbb{A}]\!]$

The problem

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

In the case of exceptions

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

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

In the case of wait-calls

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

(*i.e.* no wait call was invoked)

Monads unlocked

The analysis we did in the previous slides naturally leads to the notion of a monad

Definition

A monad $(T, \eta, (-)^\star)$ is as triple such that T is a set-constructor, η is a function $\eta_X : X \to TX$ for each set X, and $(-)^\star$ is an operation

$$\frac{f : X \to TY}{f^\star : TX \to TY}$$

such that the following laws are respected: $\eta^\star = \mathrm{id}$, $f^\star \cdot \eta = f$, $(f^\star \cdot g)^\star = f^\star \cdot g^\star$

The laws above are required to forbid "weird" equations between programs

Exercise

Show that the set-constructor

$$X \mapsto X + 1$$

can be equipped with a monadic structure

Show that the set-constructor

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

can be equipped with a monadic structure

A **very** simple language of wait-calls and its semantics

$$\frac{x_i : \mathbb{A} \in \Gamma}{[\![\Gamma \vdash \texttt{return}\, x_i]\!] = \eta \cdot \pi_i} \qquad \frac{}{[\![\Gamma \vdash \texttt{return}\ *\ ]\!] = \eta \cdot !}$$

$$\frac{[\![\Gamma \vdash M : \mathbb{A}]\!] = f \qquad [\![x : \mathbb{A} \vdash N : \mathbb{B}]\!] = g}{[\![\Gamma \vdash x \leftarrow M \,;\, N : \mathbb{B}]\!] = g^{\star} \cdot f}$$

$$\frac{[\![\Gamma \vdash M : \mathbb{A}]\!] = f}{[\![\Gamma \vdash \mathrm{wait}_{\mathrm{n}}(M) : \mathbb{A}]\!] = ((d, x) \mapsto (d + n, x)) \cdot f}$$

Exercise

Show (using the inference rules) that the equations below hold.

$$[\![x \leftarrow \mathtt{return}\ * \ ; (\mathtt{return}\, x)]\!] = [\![\mathtt{return}\ *\ ]\!]$$

(hint: one of the monad laws)

$$[\![x \leftarrow \mathrm{wait}_1(\mathtt{return}\, *) \ ; (\mathtt{return}\, x)]\!] = [\![x \leftarrow \mathtt{return}\ * \ ; \mathrm{wait}_1(\mathtt{return}\, x)]\!]$$

(hint: two of the monad laws)

$$[\![x \leftarrow \mathrm{wait}_1(\mathtt{return}\, *) \ ; \mathrm{wait}_1(\mathtt{return}\, x)]\!] = [\![x \leftarrow \mathrm{wait}_2(\mathtt{return}\, *) \ ; (\mathtt{return}\, x)]\!]$$

(hint: recall the theory of wait calls)

Empowering the language

The previous language has some limitations

Empowering the language

The previous language has some limitations

There are no higher-order features

$$\frac{\Gamma, x : \mathbb{A} \vdash V : \mathbb{B}}{\Gamma \vdash \lambda x : \mathbb{A}.\, V : \mathbb{A} \to \mathbb{B}} \qquad \frac{\Gamma \vdash V : \mathbb{A} \to \mathbb{B} \quad \Gamma \vdash U : \mathbb{A}}{\Gamma \vdash V\, U : \mathbb{B}}$$

Empowering the language

The previous language has some limitations

There are no higher-order features

$$\frac{\Gamma, x : \mathbb{A} \vdash V : \mathbb{B}}{\Gamma \vdash \lambda x : \mathbb{A}.\, V : \mathbb{A} \to \mathbb{B}} \qquad \frac{\Gamma \vdash V : \mathbb{A} \to \mathbb{B} \quad \Gamma \vdash U : \mathbb{A}}{\Gamma \vdash V\, U : \mathbb{B}}$$

There is no pairing rule

$$\frac{\Gamma \vdash M : \mathbb{A} \qquad \Gamma \vdash N : \mathbb{B}}{\Gamma \vdash \langle M, N \rangle : \mathbb{A} \times \mathbb{B}}$$

In the latter case
$\langle [\![\Gamma \vdash M : \mathbb{A}]\!], [\![\Gamma \vdash N : \mathbb{B}]\!] \rangle : [\![\Gamma]\!] \to T[\![\mathbb{A}]\!] \times T[\![\mathbb{B}]\!] \neq T([\![\mathbb{A}]\!] \times [\![\mathbb{B}]\!])$

Empowering the language

Solution

Strictly distinguish between effect-free values and effectful values

In other words, interpret some λ-terms as

$$[\![\Gamma \vdash V : \mathbb{A}]\!] : [\![\Gamma]\!] \longrightarrow [\![\mathbb{A}]\!]$$

and other λ-terms as

$$[\![\Gamma \vdash_{\mathsf{c}} M : \mathbb{A}]\!] : [\![\Gamma]\!] \longrightarrow T[\![\mathbb{A}]\!]$$

This requires a careful rewriting of the rules for deriving λ-terms

Semantics for effectful simply-typed λ-calculus

Types $\mathbb{A}$ are interpreted as sets $[\![\mathbb{A}]\!]$

$$[\![1]\!] = \{\star\}$$
$$[\![\mathbb{A} \times \mathbb{B}]\!] = [\![\mathbb{A}]\!] \times [\![\mathbb{B}]\!]$$
$$[\![\mathbb{A} \to \mathbb{B}]\!] = (T[\![\mathbb{B}]\!])^{[\![\mathbb{A}]\!]}$$

A typing context Γ is interpreted as

$$[\![\Gamma]\!] = [\![x_1 : \mathbb{A}_1 \times \cdots \times x_n : \mathbb{A}_n]\!] = [\![\mathbb{A}_1]\!] \times \cdots \times [\![\mathbb{A}_n]\!]$$

A higher-order language of wait calls

$$\frac{x_i : \mathbb{A} \in \Gamma}{[\![\Gamma \vdash x_i]\!] = \pi_i} \qquad \frac{}{[\![\Gamma \vdash *]\!] = !} \qquad \frac{[\![\Gamma \vdash V : \mathbb{A}]\!] = f \qquad [\![\Gamma \vdash U : \mathbb{B}]\!] = g}{[\![\Gamma \vdash \langle V, U \rangle : \mathbb{A} \times \mathbb{B}]\!] = \langle f, g \rangle}$$

$$\frac{[\![\Gamma, x : \mathbb{A} \vdash_{\mathsf{c}} M : \mathbb{B}]\!] = f}{[\![\Gamma \vdash \lambda x : \mathbb{A}.\, M : \mathbb{A} \to \mathbb{B}]\!] = \lambda f} \qquad \frac{[\![\Gamma \vdash V : \mathbb{A} \times \mathbb{B}]\!] = f}{[\![\Gamma \vdash \pi_1 V : \mathbb{A}]\!] = \pi_1 \cdot f}$$

. . . .

$$\frac{[\![\Gamma \vdash V : \mathbb{A}]\!] = f}{[\![\Gamma \vdash_{\mathsf{c}} \texttt{return}\ V : \mathbb{A}]\!] = \eta \cdot f} \qquad \frac{[\![\Gamma \vdash_{\mathsf{c}} M : \mathbb{A}]\!] = f \qquad [\![x : \mathbb{A} \vdash_{\mathsf{c}} N : \mathbb{B}]\!] = g}{[\![\Gamma \vdash_{\mathsf{c}} x \leftarrow M\, ;\, N : \mathbb{B}]\!] = g^{\star} \cdot f}$$

$$\frac{[\![\Gamma \vdash V : \mathbb{A} \to \mathbb{B}]\!] = f \quad [\![\Gamma \vdash U : \mathbb{A}]\!] = g}{[\![\Gamma \vdash_{\mathsf{c}} V\, U : \mathbb{B}]\!] = \mathrm{app} \cdot \langle f, g \rangle}$$

$$\frac{[\![\Gamma \vdash_{\mathsf{c}} M : \mathbb{A}]\!] = f}{[\![\Gamma \vdash_{\mathsf{c}} \mathrm{wait_n}(M) : \mathbb{A}]\!] = ((d, x) \mapsto (d + n, x)) \cdot f}$$

Exercises

Build a λ-term that receives a value, waits one second, and returns the same value. Run this in `Haskell` using `Code.hs`. What is the value obtained when you feed this function with "Hi"? Justify.

Can you build a λ-term that receives a function $f : \mathbb{A} \to \mathbb{A}$, receives a value $x : \mathbb{A}$, and applies f to x twice? In classical λ-calculus such would be defined as

$$\lambda f : \mathbb{A} \to \mathbb{A}, \lambda x : \mathbb{A}.\ f(f\ x)$$

Sharing contexts

It is very useful to have two programs M, N in sequential composition $x \leftarrow M \,;\, N$ that are able share contexts

In other words, it would be useful to have the following rule for sequential composition

$$\frac{\Gamma \vdash_{\mathsf{c}} M : \mathbb{A} \qquad \Gamma, x : \mathbb{A} \vdash_{\mathsf{c}} N : \mathbb{B}}{\Gamma \vdash_{\mathsf{c}} x \leftarrow M \,;\, N : \mathbb{B}}$$

Sharing contexts

It is very useful to have two programs M, N in sequential composition $x \leftarrow M \,;\, N$ that are able share contexts

In other words, it would be useful to have the following rule for sequential composition

$$\frac{\Gamma \vdash_{\mathsf{c}} M : \mathbb{A} \qquad \Gamma, x : \mathbb{A} \vdash_{\mathsf{c}} N : \mathbb{B}}{\Gamma \vdash_{\mathsf{c}} x \leftarrow M \,;\, N : \mathbb{B}}$$

This would allow us to solve the previous exercise quite easily

$$\lambda f : \mathbb{A} \to \mathbb{A}, \lambda x : \mathbb{A}.\, y \leftarrow f(x) \,;\, f(y)$$

Sharing contexts

The natural way of interpreting the rule would be

$$\frac{[\![\Gamma \vdash_{\mathsf{c}} M : \mathbb{A}]\!] = f \qquad [\![\Gamma, x : \mathbb{A} \vdash_{\mathsf{c}} N : \mathbb{B}]\!] = g}{[\![\Gamma \vdash_{\mathsf{c}} x \leftarrow M \,;\, N : \mathbb{B}]\!] = g^{\star} \cdot \langle \mathrm{id}, f \rangle}$$

but $\langle \mathrm{id}, f \rangle : [\![\Gamma]\!] \longrightarrow [\![\Gamma]\!] \times T[\![\mathbb{A}]\!]$ and $g^{\star} : T([\![\Gamma]\!] \times [\![\mathbb{A}]\!]) \longrightarrow T[\![\mathbb{B}]\!]$

We need to find a suitable function

$$\mathrm{str} : [\![\Gamma]\!] \times T[\![\mathbb{A}]\!] \longrightarrow T([\![\Gamma]\!] \times [\![\mathbb{A}]\!])$$

Sharing contexts

The natural way of interpreting the rule would be

$$\frac{[\![\Gamma \vdash_{\mathsf{c}} M : \mathbb{A}]\!] = f \qquad [\![\Gamma, x : \mathbb{A} \vdash_{\mathsf{c}} N : \mathbb{B}]\!] = g}{[\![\Gamma \vdash_{\mathsf{c}} x \leftarrow M \, ; N : \mathbb{B}]\!] = g^{\star} \cdot \langle \mathrm{id}, f \rangle}$$

but $\langle \mathrm{id}, f \rangle : [\![\Gamma]\!] \longrightarrow [\![\Gamma]\!] \times T[\![\mathbb{A}]\!]$ and $g^{\star} : T([\![\Gamma]\!] \times [\![\mathbb{A}]\!]) \longrightarrow T[\![\mathbb{B}]\!]$

We need to find a suitable function

$$\mathrm{str} : [\![\Gamma]\!] \times T[\![\mathbb{A}]\!] \longrightarrow T([\![\Gamma]\!] \times [\![\mathbb{A}]\!])$$

There is a natural way of doing this!

Tensorial strength

For every monad T and function $f : X \to Y$ we can build a function

$$Tf = (\eta \cdot f)^\star : TX \to TY$$

Note also that for every $x \in X$ we can define

$$\mathrm{id}_x : Y \to X \times Y, \quad y \mapsto (x, y)$$

From these, we define the so-called strength of T

$$\mathrm{str} : X \times TY \to T(X \times Y), \quad (x, t) \mapsto (T\mathrm{id}_x)(t)$$

Finally,

$$\frac{[\![\Gamma \vdash_c M : \mathbb{A}]\!] = f \qquad [\![\Gamma, x : \mathbb{A} \vdash_c N : \mathbb{B}]\!] = g}{[\![\Gamma \vdash_c x \leftarrow M ; N : \mathbb{B}]\!] = g^\star \cdot \mathrm{str} \cdot \langle \mathrm{id}, f \rangle}$$

Exercises

Given an explicit definition for the tensorial strength of

- the monad of exceptions,
- the monad of durations

Consider the λ-terms

$$\lambda f : \mathbb{A} \to \mathbb{A}, \lambda x : \mathbb{A}.\, y \leftarrow f(x) \,;\, f(y)$$

$$g = \lambda x : \mathbb{A}.\, \mathrm{wait}_1(\texttt{return}\ x)$$

What is the result of computing the λ-term below?

$$\Big(\lambda f : \mathbb{A} \to \mathbb{A}, \lambda x : \mathbb{A}.\, y \leftarrow f(x) \,;\, f(y)\Big)\ g\ \text{"}Hi\text{"}$$

Table of Contents

Going generic

Let us generalise what we learned about wait calls to arbitrary algebraic effects. We choose an algebraic theory (Σ, E) and obtain

$$\frac{x_i : \mathbb{A} \in \Gamma}{\Gamma \vdash x_i : \mathbb{A}} \qquad \frac{}{\Gamma \vdash * : 1} \qquad \frac{\Gamma \vdash V : \mathbb{A} \qquad \Gamma \vdash U : \mathbb{B}}{\Gamma \vdash \langle V, U \rangle : \mathbb{A} \times \mathbb{B}}$$

$$\frac{\Gamma, x : \mathbb{A} \vdash_{\mathsf{c}} M : \mathbb{B}}{\Gamma \vdash \lambda x : \mathbb{A}.\, M : \mathbb{A} \to \mathbb{B}} \qquad \frac{\Gamma \vdash V : \mathbb{A} \times \mathbb{B}}{\Gamma \vdash \pi_1 V : \mathbb{A}}$$

. . . .

$$\frac{\Gamma \vdash V : \mathbb{A}}{\Gamma \vdash_{\mathsf{c}} \texttt{return}\ V : \mathbb{A}} \qquad \frac{\Gamma \vdash_{\mathsf{c}} M : \mathbb{A} \qquad \Gamma, x : \mathbb{A} \vdash_{\mathsf{c}} N : \mathbb{B}}{\Gamma \vdash_{\mathsf{c}} x \leftarrow M\, ;\, N : \mathbb{B}}$$

$$\frac{\Gamma \vdash V : \mathbb{A} \to \mathbb{B} \quad \Gamma \vdash U : \mathbb{A}}{\Gamma \vdash_{\mathsf{c}} V\, U : \mathbb{B}} \qquad \frac{\sigma : n \in \Sigma \qquad \forall i \leq n.\, \Gamma \vdash_{\mathsf{c}} M_i : \mathbb{A}}{\Gamma \vdash_{\mathsf{c}} \sigma(M_1, \ldots M_n) : \mathbb{A}}$$

Going generic

We now need to choose a suitable monad T to interpret the language

There are sophisticated ways of doing this

It is even possible to automatically generate a monad for the language

Here we will simply choose monads that seem suitable for the job. By suitable, we mean that for every set X the set TX must be a (Σ, E)-algebra.

A generic semantics

$$\frac{x_i : \mathbb{A} \in \Gamma}{[\![\Gamma \vdash x_i]\!] = \pi_i} \qquad \frac{}{[\![\Gamma \vdash *]\!] = !} \qquad \frac{[\![\Gamma \vdash V : \mathbb{A}]\!] = f \qquad [\![\Gamma \vdash U : \mathbb{B}]\!] = g}{[\![\Gamma \vdash \langle V, U\rangle : \mathbb{A} \times \mathbb{B}]\!] = \langle f, g\rangle}$$

$$\frac{[\![\Gamma, x : \mathbb{A} \vdash_{\mathsf{c}} M : \mathbb{B}]\!] = f}{[\![\Gamma \vdash \lambda x : \mathbb{A}.\, M : \mathbb{A} \to \mathbb{B}]\!] = \lambda f} \qquad \frac{[\![\Gamma \vdash V : \mathbb{A} \times \mathbb{B}]\!]}{[\![\Gamma \vdash \pi_1 V : \mathbb{A}]\!] = \pi_1 \cdot f}$$

. . . .

$$\frac{[\![\Gamma \vdash V : \mathbb{A}]\!] = f}{[\![\Gamma \vdash_{\mathsf{c}} \mathtt{return}\ V : \mathbb{A}]\!] = \eta \cdot f} \qquad \frac{[\![\Gamma \vdash_{\mathsf{c}} M : \mathbb{A}]\!] = f \qquad [\![\Gamma, x : \mathbb{A} \vdash_{\mathsf{c}} N : \mathbb{B}]\!] = g}{[\![\Gamma \vdash_{\mathsf{c}} x \leftarrow M\, ;\, N : \mathbb{B}]\!] = g^\star \cdot \mathrm{str} \cdot f}$$

$$\frac{[\![\Gamma \vdash V : \mathbb{A} \to \mathbb{B}]\!] = f \quad [\![\Gamma \vdash U : \mathbb{A}]\!] = g}{[\![\Gamma \vdash_{\mathsf{c}} V\ U : \mathbb{B}]\!] = \mathrm{app} \cdot \langle f, g\rangle} \qquad \frac{\sigma : n \in \Sigma \qquad \forall i \le n.\ [\![\Gamma \vdash_{\mathsf{c}} M_i : \mathbb{A}]\!] = f_i}{[\![\Gamma \vdash_{\mathsf{c}} \sigma(M_1, \dots M_n)]\!] = [\![\sigma]\!] \cdot \langle f_1, \dots, f_n\rangle}$$