INSTITUT
DE RECHERCHE
EN INFORMATIQUE
FONDAMENTALE

Pierre Cagne

for Kan Extension Seminar II

Université Paris Diderot

When computational monads go clubbing

Category Theory 2017 – Vancouver

1. Crash course in computational monads

2. Clubs

3. Strong monads

4. Computational monads as clubs

1

Crash course in computational monads

Program

```
def f():
  x = input(”Enter a number:”)
  return 2*int(x)
def g(y):
  return y*y

a = g(f())
b = g(f())
```

Modelization

Program

```
def f():
  x = input(”Enter a number:”)
  return 2*int(x)
def g(y):
  return y*y

a = g(f())
b = g(f())
```

Modelization

$$f : 1 \to \mathbb{N}$$

$$g : \mathbb{N} \to \mathbb{N}$$

Hence f is just a constant integer $n \in \mathbb{N}$, and we should get $a = b = g(n)$...

Program

```
def f():
  x = input("Enter a number:")
  return 2*int(x)
def g(y):
  return y*y

a = g(f())
b = g(f())
```

Modelization

$$f : 1 \to \mathbb{N}^{\mathbb{N}}$$

$$g : \mathbb{N} \to \mathbb{N}$$

Now f is not constant anymore.

Program

```
def f():
  x = input("Enter a number:")
  return 2*int(x)
def g(y):
  return y*y

a = g(f())
b = g(f())
```

Modelization

$$f : 1 \to \mathbb{N}^{\mathbb{N}}$$

$$g : \mathbb{N} \to \mathbb{N}$$

Now f is not constant anymore.
But how to compose g with f now?

Program

```python
def f():
  x = input("Enter a number:")
  return 2*int(x)
def g(y):
  return y*y

a = g(f())
b = g(f())
```

Modelization

$$f : 1 \to \mathbb{N}^{\mathbb{N}}$$

$$g : \mathbb{N} \to \mathbb{N}$$

Now f is not constant anymore.
But how to compose g with f now?

Use Kleisli composition of the monad

$$(-)^{\mathbb{N}} : \mathsf{Set} \to \mathsf{Set}$$

Program

```
def f(a,b):
  if b == 0: raise Error
  else: return a/b
def g(y):
  return y*y

a = g(f(4,0))
b = g(f(4,2))
```

Modelization

Program

```
def f(a,b):
  if b == 0: raise Error
  else: return a/b
def g(y):
  return y*y

a = g(f(4,0))
b = g(f(4,2))
```

Modelization

$$f : \mathbb{N}^2 \to \mathbb{N}$$

$$g : \mathbb{N} \to \mathbb{N}$$

Hence *a* as well as *b* should have a defined value in $\mathbb{N}$...

Program

```
def f(a,b):
  if b == 0: raise Error
  else: return a/b
def g(y):
  return y*y

a = g(f(4,0))
b = g(f(4,2))
```

Modelization

$$f : \mathbb{N}^2 \to \mathbb{N} + \mathbf{e}$$

$$g : \mathbb{N} \to \mathbb{N}$$

$\mathbf{e}$ is a singleton
Now f is only partially defined.

Program

```
def f(a,b):
  if b == 0: raise Error
  else: return a/b
def g(y):
  return y*y

a = g(f(4,0))
b = g(f(4,2))
```

Modelization

$$f : \mathbb{N}^2 \to \mathbb{N} + \mathbf{e}$$

$$g : \mathbb{N} \to \mathbb{N}$$

$\mathbf{e}$ is a singleton
Now f is only partially defined.
But how to compose g with f now?

Program

```
def f(a,b):
  if b == 0: raise Error
  else: return a/b
def g(y):
  return y*y

a = g(f(4,0))
b = g(f(4,2))
```

Modelization

$$f : \mathbb{N}^2 \to \mathbb{N} + \mathbf{e}$$

$$g : \mathbb{N} \to \mathbb{N}$$

$\mathbf{e}$ is a singleton
Now f is only partially defined.
But how to compose g with f now?

Use Kleisli composition of the monad

$$- + \mathbf{e} : \mathrm{Set} \to \mathrm{Set}$$

Big picture (Moggi)

Side effects of a programming language can be encoded by a monad T.

Big picture (Moggi)

Side effects of a programming language can be encoded by a monad T.

Distinguish values (objects A) from computations (images TA).

Big picture (Moggi)

Side effects of a programming language can be encoded by a monad T.

Distinguish values (objects A) from computations (images TA).

Programs are interpreted by morphisms $A \to TB$.

Composition of programs occurs in the Kleisli category of T.

Big picture (Moggi)

Side effects of a programming language can be encoded by a monad T.

Distinguish values (objects A) from computations (images TA).

Programs are interpreted by morphisms $A \to TB$.

Composition of programs occurs in the Kleisli category of T.

(One wants good properties for T: strong, pullbacks preserving, etc.)

2

Clubs

Clubs

$\mathcal{M}$: cartesian natural transformations in $[\mathcal{A}, \mathcal{A}]$.

Clubs

$\mathcal{M}$: cartesian natural transformations in $[\mathcal{A}, \mathcal{A}]$.

$\alpha : T \to S$ is cartesian when each

$$\begin{tikzcd} TX \arrow[r] \arrow[d] & SX \arrow[d] \\ TX \arrow[r] & SY \end{tikzcd}$$

is a pullback square.

Clubs

$\mathcal{M}$: cartesian natural transformations in $[\mathcal{A}, \mathcal{A}]$.

Definition (Clubs)

A monad (S, j, n) on $\mathcal{A}$ is a club whenever $\mathcal{M}/S$ is monoidal for:

$$(T \xrightarrow{\alpha} S) \otimes (T' \xrightarrow{\beta} S) = TT' \xrightarrow{\alpha\beta} SS \xrightarrow{n} S$$

Clubs

$\mathcal{M}$: cartesian natural transformations in $[\mathcal{A}, \mathcal{A}]$.

Definition (Clubs)

A monad (S, j, n) on $\mathcal{A}$ is a club whenever $\mathcal{M}/S$ is monoidal for:

$$(T \xrightarrow{\alpha} S) \otimes (T' \xrightarrow{\beta} S) = TT' \xrightarrow{\alpha\beta} SS \xrightarrow{n} S$$

Remark: in particular, cartesian monads are clubs.

Clubs

$\mathcal{M}$: cartesian natural transformations in $[\mathcal{A}, \mathcal{A}]$.

Definition (Clubs)

A monad (S, j, n) on $\mathcal{A}$ is a club whenever $\mathcal{M}/S$ is monoidal for:

$$(T \xrightarrow{\alpha} S) \otimes (T' \xrightarrow{\beta} S) = TT' \xrightarrow{\alpha\beta} SS \xrightarrow{n} S$$

Remark: in particular, cartesian monads are clubs.

Idea

When $\mathcal{A}$ has a terminal 1, exploits the equivalence $\mathcal{A}/S1 \simeq \mathcal{M}/S$.
Clubs over S are now *easily* spotted as monoids in $\mathcal{A}/S1$.

Tensoring in $\mathcal{A}/S1$

For $K \xrightarrow{f} S1$ and $X \xrightarrow{g} S1$, the tensor $f \otimes g$ is obtained as:

$$
\begin{tikzcd}
K \times_{S1} SX \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & K \arrow[d, "f"] \\
SX \arrow[r] \arrow[d, "S(g)"] & S1 \\
SS1 \arrow[d, "n_1"] & \\
S1 &
\end{tikzcd}
$$

Tensoring in $\mathcal{A}/S1$

For $K \xrightarrow{f} S1$ and $X \xrightarrow{g} S1$, the tensor $f \otimes g$ is obtained as:

$$\begin{tikzcd}
K \times_{S1} SX \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & K \arrow[d, "f"] \\
SX \arrow[r] \arrow[d, "S(g)"] & S1 \\
SS1 \arrow[d, "n_1"] & \\
S1 &
\end{tikzcd}$$

Warning

Highly non symmetric!

Remark

Reminiscent of the operadic substitution product.

Enriched clubs

$\mathcal{V}$: "nice" cartesian closed category

Enriched clubs

$\mathcal{V}$: "nice" cartesian closed category

Definition

A $\mathcal{V}$-monad (S, j, n) on a $\mathcal{V}$-category $\mathcal{A}$ is a enriched club whenever (S_0, j, n) is an ordinary one on $\mathcal{A}_0$.

Enriched clubs

$\mathcal{V}$: "nice" cartesian closed category

Definition

A $\mathcal{V}$-monad (S, j, n) on a $\mathcal{V}$-category $\mathcal{A}$ is a enriched club whenever (S_0, j, n) is an ordinary one on $\mathcal{A}_0$.

Key feature

There is still a one-to-one correspondance between clubs over S and monoids in $\mathcal{A}_0/S_0 1$.

3

Strong monads

Every category is canonically enriched

Fact

Every small category $\mathcal{A}$ with products is enriched over $\mathcal{V} = \mathrm{Psh}(\mathcal{A})$, by defining

$$\mathcal{A}(A, B) : C \mapsto \mathcal{A}(A \times C, B)$$

Every category is canonically enriched

Fact

Every small category $\mathcal{A}$ with products is enriched over $\mathcal{V} = \mathrm{Psh}(\mathcal{A})$, by defining

$$\mathcal{A}(A, B) : C \mapsto \mathcal{A}(A \times C, B)$$

A $\mathcal{V}$-monad is then an ordinary monad (T_0, j, n) on $\mathcal{A}$ together with a natural map $\sigma_{A,C} : SA \times C \to S(A \times C)$ that makes T_0 a strong monad.

Every category is canonically enriched

Fact

Every small category $\mathcal{A}$ with products is enriched over $\mathcal{V} = \mathsf{Psh}(\mathcal{A})$, by defining

$$\mathcal{A}(A, B) : C \mapsto \mathcal{A}(A \times C, B)$$

A $\mathcal{V}$-monad is then an ordinary monad (T_0, j, n) on $\mathcal{A}$ together with a natural map $\sigma_{A,C} : SA \times C \to S(A \times C)$ that makes T_0 a strong monad.

Conclusion

Cartesian strong monads are enriched clubs. Conversely, $\mathcal{V}$-clubs are *good enough* to be effects.

4

Computational monads as clubs

Clubs over Error

Take $\mathcal{A} = \mathsf{Set}$ for the following.

Clubs over Error

Take $\mathcal{A} = \mathrm{Set}$ for the following.

The monad $S = - + \mathbf{e} : \mathrm{Set} \to \mathrm{Set}$ is strong and cartesian.

Clubs over Error

Take $\mathcal{A} = \mathrm{Set}$ for the following.

The monad $S = - + \mathbf{e} : \mathrm{Set} \to \mathrm{Set}$ is strong and cartesian.

Hence it is an enriched club, and clubs over S are easily spotted as monoids in $\mathrm{Set}/1 + \mathbf{e}$.

Clubs over Error

Take $\mathcal{A} = \mathsf{Set}$ for the following.

The monad $S = - + \mathbf{e} : \mathsf{Set} \to \mathsf{Set}$ is strong and cartesian.

Hence it is an enriched club, and clubs over S are easily spotted as monoids in $\mathsf{Set}/1 + \mathbf{e}$.

Those are $M + K_{\mathbf{e}} \to 1 + \mathbf{e}$ where M is a plain monoid, which induces the club

$$T : X \mapsto (M \times X) + K_{\mathbf{e}}$$

What is this effect?

Program

Modelization

A program is modelled as a map

$$A \to TB$$

Composing programs is Kleisli-composing functions: for $f : A \to TB$ and $g : B \to TC$, define $g \circ f(x)$ as

$$f(x) \text{ if } f(x) \in K_{\mathbf{e}}$$
$$g(y) \text{ if } f(x) = (m, y) \text{ and } g(y) \in K_{\mathbf{e}}$$
$$(mm', z) \text{ if } f(x) = (m, y) \text{ and } g(y) = (m', z)$$

Here : $T = (M \times -) + K_{\mathbf{e}}$ with M monoid.

What is this effect?

Program

```
MAX = 2147483647
def f(a,b):
  print "Computing a quotient..."
  print "Div by 0 raise an error."
  if b == 0: raise DivisionByZero
  else: return a/b
def g(y):
  print "Squaring..."
  print "Too big numbers raise errors."
  if y > MAX: raise TooBigError
  else: return y*y

a = g(f(4,0))
b = g(f(2**32,2))
c = g(f(4,2))
```

Modelization

A program is modelled as a map

$$A \longrightarrow TB$$

Composing programs is Kleisli-composing functions: for $f : A \longrightarrow TB$ and $g : B \to TC$, define $g \circ f(x)$ as

$$\begin{array}{l} f(x) \text{ if } f(x) \in K_{\mathbf{e}} \\ g(y) \text{ if } f(x) = (m, y) \text{ and } g(y) \in K_{\mathbf{e}} \\ (mm', z) \text{ if } f(x) = (m, y) \text{ and } g(y) = (m', z) \end{array}$$

Here : $T = (M \times -) + K_{\mathbf{e}}$ with M the free monoid on the ASCII alphabet and $K_{\mathbf{e}} = \{e_1, e_2\}$.

Thank you!

http://www.normalesup.org/~cagne/

https://pierrecagne.github.io