Basic category theory - Monads

Gouter des doctorants

Jérémy Dubut

LSV, ENS Cachan

Friday, 3rd April, 2015

Monads (in programming languages)

monads in Kleisli triples formulation

a *type constructor* $M : Type \longrightarrow Type$

a *unit function* $return_A : A \longrightarrow M\ A$

a *binding operation*
$bind_{A,B} : M\ A \to (A \to M\ B) \to M\ B$

Exception monad

```
type M A = Val of A | Exn

let return a = Val a

let rec bind E f = match E with
| Exn -> Exn
| Val a -> f a
```

satisfying :

$$bind(return\ x, f) = f\ x$$

$$bind(x, return) = x$$

$$bind(bind(x, f), g) = bind(x, \lambda y.bind(f\ y, g))$$

Example II : list monad

- type M A = [] | x : :L of A*M A
- let return a = a : :[]
- let rec bind L f = match L with
  | [] -> []
  | x : :K -> cat (f x) (bind K f)
  where rec cat L K = match L with
  | [] -> []
  | x : :J -> x : :(cat J K)

Programming languages as categories

From a programming language (for example simply typed lambda-calculus), we can construct a category **Type** :

- $Ob(\mathbf{Type})$ is the class of types
- $\mathbf{Type}(A, B)$ is the set of programs of type $A \longrightarrow B$ modulo $\beta\eta$ equivalence
- id_A is the identity function $\lambda x.x$
- composition is the composition of programs $Q \circ P = \lambda x.Q\ (P\ x)$

Categorical view of Kleisli triples I : the type constructor functor

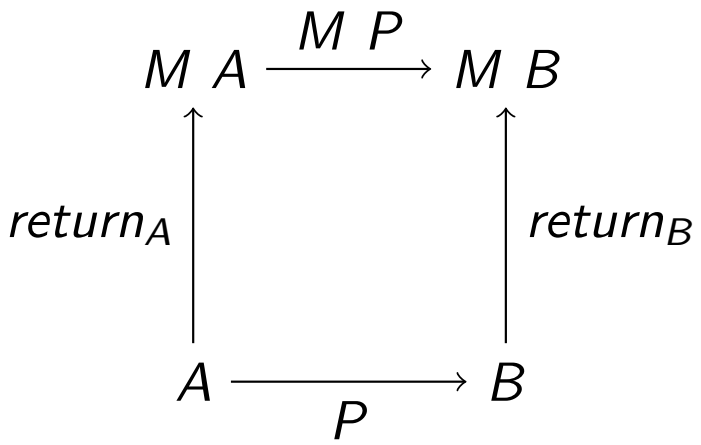
M can be extended to a functor M : **Type** $\longrightarrow$ **Type** :

$$M\ P = \lambda x : M\ A.bind(x, \lambda y : A.return\ (P\ y)) : M\ A \longrightarrow M\ B$$

with P of type $A \longrightarrow B$.

Categorical view of Kleisli triples II : the unit natural transformation

$return_A$ is a program of type $A \longrightarrow M\ A$. In fact, $return : Id \longrightarrow M$ is a natural transformation i.e. :



$$\lambda x.bind(return\ x, \lambda y.return\ (P\ y)) = \lambda x.return\ (P\ x)$$

Categorical view of Kleisli triples III : the multiplication natural transformation

We have a program of type $M\ (M\ A) \longrightarrow M\ A$:

$$\mu_A = \lambda x : M\ (M\ A).bind(x, \lambda y : M\ A.y)$$

This defines a natural transformation $\mu : M \circ M \longrightarrow M$ i.e. :

$$\begin{array}{ccc} M\ A & \xrightarrow{M\ P} & M\ B \\ {\scriptstyle \mu_A}\uparrow & & \uparrow{\scriptstyle \mu_B} \\ M\ (M\ A) & \xrightarrow[M\ (M\ P)]{} & M\ (M\ B) \end{array}$$

$$\lambda x.bind(bind(x, \lambda z.z), \lambda y.return\ (P\ y)) =$$

$$\lambda x.bind(bind(x, \lambda y.return\ bind(y, \lambda w.P\ w)), \lambda z.z)$$

Categorical view of Kleisli triples IV : the monad laws