{ Monads in Coq and Categories }

Department of Mathematics,
Statistics and Computer Science

Amir Faridi
Spring 1404

Table of contents

Introduction

- Born in **1967** by **Jean Bénabou** – in Category Theory
- Programming with **Effects** – **Eugenio Moggi**
- Contributions in Haskell – **Philip Wadler**

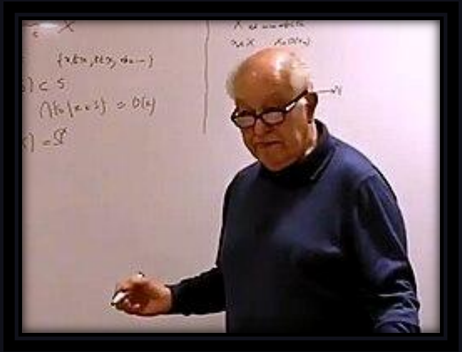


Fig1. Jean Bénabou (1932-2022), Jean Bénabou, “Fibered categories and the foundations of naïve category theory”[2]

Example (Coq)

Expr

```
Inductive expr : Type :=
| Val : nat -> expr
| Div : expr -> expr -> expr.
```

Option

```
Inductive option (A : Type) : Type :=
  | Some : A -> option A
  | None : option A.
```

Fig2. Eugenio Moggi, Moggi, Eugenio (1991). “Notions of Computation and Monads”[2]

Example (Cont.)

```
Fixpoint eval_no_monad (e : expr) : option nat :=
 match e with
 | Val n => Some n
 | Div e1 e2 =>
     match eval_no_monad e1 with
     | Some n1 =>
         match eval_no_monad e2 with
         | Some n2 =>
             if n2 =? 0 then None else Some (n1 / n2)
         | None => None
         end
     | None => None
     end
 end.
```

Idea of Monads

```
Definition ret {A : Type} (x : A) : option A := Some x.

Definition bind {A B : Type} (oa : option A) (f : A ->
   option B) : option B :=
 match oa with
 | Some a => f a
 | None => None
 end.
```

Monad in Coq

```
Fixpoint eval (e : expr) : option nat :=
 match e with
 | Val n => ret n
 | Div e1 e2 =>
     bind (eval e1) (fun n1 =>
     bind (eval e2) (fun n2 =>
     if n2 =? 0 then None else ret (n1 / n2)))
 end.
```

Monad in Coq (Cont.)

```
Class Monad@{d c} (T : Type@{d} -> Type@{c}) : Type :=
{
    ret : forall {A : Type@{d}}, A -> T A ;
    bind : forall {A B : Type@{d}}, T A -> (A -> T B) -> T B
}.
```

Monads – Category Theory

[1]Definition. A monad on a category $\mathbb{C}$ consists of an endofunctor $T : \mathbb{C} \to \mathbb{C}$, natural transformations $\eta : 1 \Rightarrow T$, and $\mu : T^2 \Rightarrow T$ satisfying the two commutative diagrams below, that is:

$$\begin{tikzcd} T^3 \arrow[r, "T\mu"] \arrow[d, "\mu_T"'] & T^2 \arrow[d, "\mu"] \\ T^2 \arrow[r, "\mu"'] & T \end{tikzcd}$$

$$\begin{tikzcd} T \arrow[r, "\eta_T"] \arrow[dr, "="'] & T^2 \arrow[d, "\mu"] & T \arrow[l, "T\eta"'] \arrow[dl, "="] \\ & T & \end{tikzcd}$$

$\mu \circ \mu_T = \mu \circ T\mu$

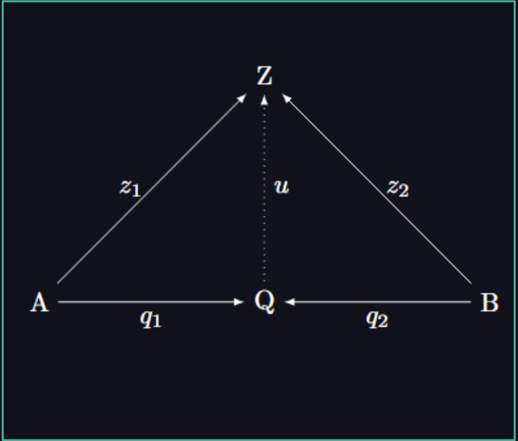
$\mu \circ \eta_T = 1 = \mu \circ T\eta$

Coproduct

[1]Definition. $(Q, q_1 : A \to Q, q_2 : B \to Q)$ is a coproduct of A and B, if

$$\forall\,(Z, z_1 : A \to Z\,, z_2 : B \to Z)\ \exists!\, u : Q \to Z$$

$$s.t.\ u \circ q_i = z_i, \qquad i = 1, 2$$



Coproduct Diagram

Coproduct in Sets: Disjoint Union

Example

Endofunctor: $T : Sets \to Sets$, where $TA = A + 1$

Natural Transformation: $\eta_A : A \to A + 1$, where $\eta_A = i_1$

Natural Transformation: $\mu_A : (A + 1) + 1 \to A + 1$, where $\mu_A = [id_{A+1}, i_2]$

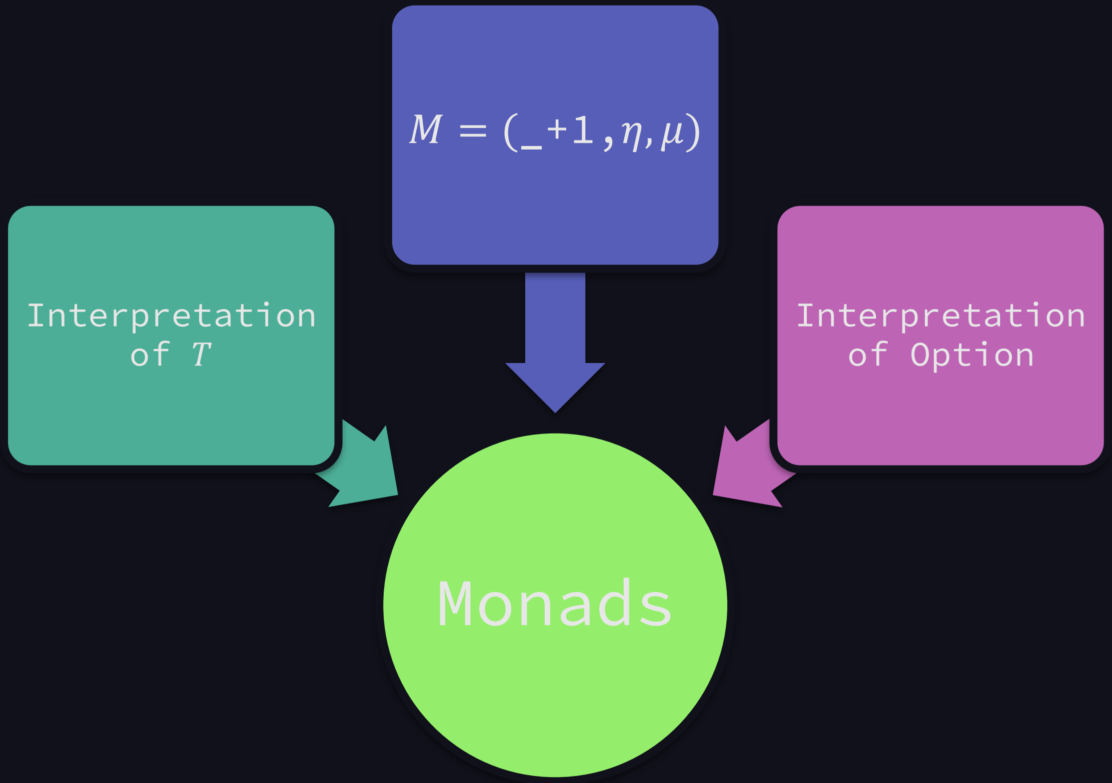
Kleisli Category

[3]**Definition.** For any monad $M = \langle T, \eta, \mu \rangle$ on $\mathbb{C}$, the Kleisli Category $Kl(M)$ is defined as follows:

- Objects: $A_T, B_T, \dots$
- Morphisms: $f: A \to TB \in \mathbb{C}_1$
- $id(A_T) : \eta_A : A \to TA$
- Composition: For $f_T: A_T \to B_T$ and $g_T: B_T \to C_T$:

$$
\begin{tikzcd}
A \arrow[r, "g_T \circ f_T"] \arrow[d, "f_T"'] & TC \\
TB \arrow[r, "Tg_T"'] & TTC \arrow[u, "\mu_C"']
\end{tikzcd}
$$

Connection


$M = (\_+1, \eta, \mu)$
Interpretation of T
Interpretation of Option
Monads

References

1. *Awodey, Steve. "Category Thoery". 2nd ed., Oxford University Press, 2010*
2. *All portraits and personal images in this presentation are taken from publicly available content on Wikipedia pages of the respective individuals.*

Thank you