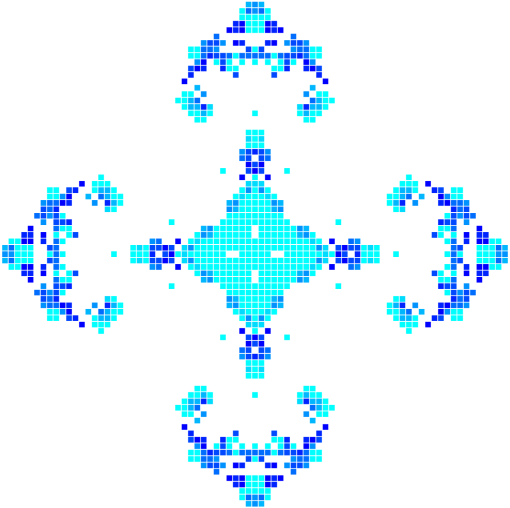
Monades, Comonades et Automates cellulaires

Jérémy S. Cochoy

INRIA Paris-Saclay

Octobre 2015



Les types

Qu'est-ce qu'un type ?

C'est un *ensemble* de valeurs.

Examples :

- $Int = \{-2147483648, \ldots, 2147483647\}$
- $Bool = \{True, False\}$
- $Char = \{'a', 'b', 'c', \ldots\}$
- $[Bool] = \{[], [True], [False], [True, False], [False, True], \ldots\}$
- $[a]$

Les types

Qu'est-ce qu'un type ?

C'est un *ensemble* de valeurs.

Examples :

- $Int = \{-2147483648, \ldots, 2147483647\}$
- $Bool = \{True, False\}$
- $Char = \{'a', 'b', 'c', \ldots\}$
- $[Bool] = \{[], [True], [False], [True, False], [False, True], \ldots\}$
- $[a]$

Les types

Qu'est-ce qu'un type ?

C'est un *ensemble* de valeurs.

Examples :

- $Int = \{-2147483648, \ldots, 2147483647\}$
- $Bool = \{True, False\}$
- $Char = \{'a', 'b', 'c', \ldots\}$
- $[Bool] = \{[], [True], [False], [True, False], [False, True], \ldots\}$
- $[a]$

Les types

Qu'est-ce qu'un type ?

C'est un *ensemble* de valeurs.

Examples :

- $Int = \{-2147483648, \ldots, 2147483647\}$
- $Bool = \{True, False\}$
- $Char = \{'a', 'b', 'c', \ldots\}$
- $[Bool] = \{[], [True], [False], [True, False], [False, True], \ldots\}$
- $[a]$

Les types

Construire son type :

- Trival = Plus | Minus | Zero
- Box a = InABox a
- Maybe a = Just a | Nothing
- Either a b = Left a | Right b

Just, Nothing, InABox etc portent le doux nom de *constructeur de type*. C'est aussi le cas de *[]*.

Les types

Construire son type :

- Trival = Plus | Minus | Zero
- Box a = InABox a
- Maybe a = Just a | Nothing
- Either a b = Left a | Right b

Just, Nothing, InABox etc portent le doux nom de *constructeur de type*. C'est aussi le cas de *[]*.

Les types

Construire son type :

- Trival = Plus | Minus | Zero
- Box a = InABox a
- Maybe a = Just a | Nothing
- Either a b = Left a | Right b

Just, Nothing, InABox etc portent le doux nom de *constructeur de type*. C'est aussi le cas de *[]*.

Les types

Construire son type :

- Trival = Plus | Minus | Zero
- Box a = InABox a
- Maybe a = Just a | Nothing
- Either a b = Left a | Right b

Just, Nothing, InABox etc portent le doux nom de *constructeur de type*. C'est aussi le cas de *[]*.

Les types

Construire son type :

- Trival = Plus | Minus | Zero
- Box a = InABox a
- Maybe a = Just a | Nothing
- Either a b = Left a | Right b

Just, Nothing, InABox etc portent le doux nom de *constructeur de type*. C'est aussi le cas de *[]*.

Les types : Les fonctions

Une fonction a aussi un type :

- square : : Float -> Float
- floor : : Float -> Int
- showInt : : Int -> String
- (+) : : Int -> Int -> Int
- id : : a -> a

Les fonctions

Une fonction a aussi un type :

- Une fonction a un type de la forme *a* -> *b*
- (+) : : (Int) -> (Int -> Int)
- (5+) : : Int -> Int
- map : : (a -> b) -> [a] -> [b]

Les fonctions

Ca se compose

- f1 : : a -> b
- f2 : : b -> c
- f2 . f1 : : a -> c
- . : : (a -> b) -> (b -> c) -> (a -> c)

Pour l'anecdote, la collection de tous les types forme une catégorie où les flèches sont les fonctions implémentables. On l'appelle la catégorie des types.

Les fonctions

Ca se compose

- f1 : : a -> b
- f2 : : b -> c
- f2 . f1 : : a -> c
- . : : (a -> b) -> (b -> c) -> (a -> c)

Pour l'anecdote, la collection de tous les types forme une catégorie où les flèches sont les fonctions implémentables. On l'appelle la catégorie des types.

Les fonctions

Ca se compose

- f1 : : a -> b
- f2 : : b -> c
- f2 . f1 : : a -> c
- . : : (a -> b) -> (b -> c) -> (a -> c)

Pour l'anecdote, la collection de tous les types forme une catégorie où les flèches sont les fonctions implémentables. On l'appelle la catégorie des types.

Les fonceurs

Un foncteur F agit sur les types ...

- a => F a

- a => Maybe a
- a => [a]

... et sur les fonctions

- a -> b => F a -> F b

- fmap square : : F Float -> F Float
- fmap id : : F a -> F a

Les foncteurs

Un foncteur F agit sur les types ...

- a => F a

- a => Maybe a
- a => [a]

... et sur les fonctions

- a -> b => F a -> F b

- fmap square : : F Float -> F Float
- fmap id : : F a -> F a

Un foncteur doit respecter des lois

- fmap id = id
- fmap (p . q) = (fmap p) . (fmap q)

Un foncteur est un endofoncteur de la catégorie des types.

Un foncteur doit respecter des lois

- fmap id = id
- fmap (p . q) = (fmap p) . (fmap q)

Un foncteur est un endofoncteur de la catégorie des types.

Donnée dans un contexte

Un traitement qui peux échouer

Combiner des traitements avec échec

L'opérateur bind

L'opérateur join

Monade

Monades - Catégories

Une monade (T, μ, η) est la donné d'un endofoncteur $T : C \to C$ et de deux transformations naturelles $\mu : T \circ T \to T$ et $\eta : 1_C \to T$ telles que :

$$\begin{tikzcd} T(T(T(X))) \arrow[r, "T(\mu_X)"] \arrow[d, "\mu_{T(X)}"'] & T(T(X)) \arrow[d, "\mu_X"] \\ T(T(X)) \arrow[r, "\mu_X"'] & T(X) \end{tikzcd}$$

$$\begin{tikzcd} T(X) \arrow[r, "\eta_{T(X)}"] \arrow[d, "T(\eta_X)"'] \arrow[dr, equal] & T(T(X)) \arrow[d, "\mu_X"] \\ T(T(X)) \arrow[r, "\mu_X"'] & T(X) \end{tikzcd}$$

c'est à dire $\mu \circ T\mu = \mu \circ \mu T$ et $\mu \circ T\eta = \mu \circ \eta T = 1_T$.

Dans notre cas, C est la catégorie des types : les objet sont les

Monades - Catégories

Une monade (T, μ, η) est la donné d'un endofoncteur $T : C \to C$ et de deux transformations naturelles $\mu : T \circ T \to T$ et $\eta : 1_C \to T$ telles que :

$$\begin{tikzcd}
T(T(T(X))) \arrow[r, "T(\mu_X)"] \arrow[d, "\mu_{T(X)}"'] & T(T(X)) \arrow[d, "\mu_X"] \\
T(T(X)) \arrow[r, "\mu_X"'] & T(X)
\end{tikzcd}$$

$$\begin{tikzcd}
T(X) \arrow[r, "\eta_{T(X)}"] \arrow[d, "T(\eta_X)"'] \arrow[dr, equal] & T(T(X)) \arrow[d, "\mu_X"] \\
T(T(X)) \arrow[r, "\mu_X"'] & T(X)
\end{tikzcd}$$

c'est à dire $\mu \circ T\mu = \mu \circ \mu T$ et $\mu \circ T\eta = \mu \circ \eta T = 1_T$.

Dans notre cas, C est la catégorie des types ; les objet sont les