Tarea 4

Universidad Nacional Autónoma de México
Facultad de Ciencias
Programación Declarativa

Fong Baeza Luis Fernando Yang
`fernandofong@ciencias.unam.mx`

Lezama Hernández María Ximena
`lezama@ciencias.unam.mx`

18 de Mayo 2018

***Definición* 1** *Una categoría C consiste de:*

- *Una clase de objetos llamada $Ob(C)$.*
- *Para cada par de objetos a y b en $Ob(C)$, Existe un conjunto de homomorfismos de a en b, que llamaremos flechas y que denotamos como $Hom_C(a, b)$. Si f es una flecha que está en $Hom_C(a, b)$, la representaremos con $a \xrightarrow{f} b$.*
- *Para cada a en $Ob(C)$ se define la flecha id_a tal que $a \xrightarrow{id_a} a$ y id_a esta en $Hom_C(a, a)$.*
- *Para cada terna a, b, c en $Ob(C)$, existe una operacion tal que:*

$$\circ_{a,b,c} : Hom_C(b, c) \times Hom_C(a, b) \to Hom_C(a, c)$$

la cual llamaremos "composicion" la cual cumple las siguientes propiedades:

***Asociatividad**: Sean a, b, c, d objetos en $Ob(C)$ y f en $Hom_C(a, b)$, g en $Hom_C(b, c)$, h en $Hom_C(c, d)$, se cumple que:*

$$(h \circ g) \circ f = h \circ (g \circ f)$$

***Identidad izquierda**:Sean a, b objetos en $Ob(C)$ y f en $Hom_C(a, b)$, se cumple que:*

$$id_b \circ f = f$$

***Identidad izquierda**: Sean a, b objetos en $Ob(C)$ y f en $Hom_C(a, b)$, se cumple que:*

$$f \circ id_a = f$$

***Definición* 2** *Un conjunto parcialmente ordenado (poset) es un conjunto con una relación reflexiva, antisimétrica y transitiva. Un homomorfismo de posets es una función que preserva el orden (también llamado función monótona).*
*La categoría **Poset** tiene por objetos a los posets y por flechas a las funciones monótonas.*

***Definición* 3** *Una categoría C se dice pequeña, cuando $Obj(C)$ y $C(A, B)$ son conjuntos, para cualquier par de objetos A, B en $Obj(C)$.*

***Definición* 4** *Para un par de categorías dadas C y B un **funtor** $F : C \to B$; consiste de dos funciones relacionadas adecuadamente con una función que cada $Ob(C)$ le asocia un objeto $Ob(D)$ de manera que*

$$F_{Ob} : Ob(C) \to Ob(D)$$

y para cualesquiera objetos a, b en $Ob(C)$, tenemos que

$$F_{a,b} : Hom_C(a, b) \to Hom_D(F_{Ob}(a), F_{Ob}(b))$$

***Definición* 5** *Sea* **Cat** *la categoría que tiene como objetos a categorías pequeñas y por flechas a funtores.*

***Definición* 6** *Sea F un funtor, $F : C \to D$ tenemos para cada par de objetos a, b en C una función*

$$C[a,b] \to D[F(a), F(b)],\ f \mapsto F(f)$$

Decimos que F es fiel si estas aplicaciones son inyectivas para todo a, b en C, que es pleno si son sobreyectivas y que es plenamente fiel el si son biyectivas. Los funtores plenamente fieles cumplen la función de "subobjeto" en el contexto categórico.

1. Definimos la clase $C_{\mathbb{N}}$ con las siguientes caracteristicas:

   - $Ob(C_{\mathbb{N}}) = \mathbb{N}$
   - Sean $n, m \in \mathbb{N}$, existe una flecha f tal que $n \xrightarrow{f} m$ si y solo si $n \leq m$

   Muestra que $C_{\mathbb{N}}$ es una categoría. ¿Qué hace un funtor $F : C_{\mathbb{N}} \to C_{\mathbb{N}}$?

   Demostración:
   Sea $P \in \mathbb{N}$ *un conjunto parcialmente ordenado*. Definimos P' como una *categoría pequeña* de manera que los objetos de P' son objetos de P y existe un morfismo de $p \to p'$ exactamente cuando $p \leq p'$ en P. Notemos que con esto determinamos de forma única la composición. La asignación $P \Rightarrow P'$ se extiende de manera obvia a un funtor $f : \mathbb{N} \to Cat$.

   Luego, no hay problema en abusar del lenguaje y pensar que $\mathbb{N}$ es una subcategoría de Cat (un conjunto parcialmente ordenado "es" una categoría). Recíprocamente, dada una categoría pequeña C, definimos el conjunto ordenado UC como el conjunto de objetos C_0 de C con el siguiente orden: $a \leq b$ en UC si y sólo si $C[a,b] \neq \emptyset$. Se verifica inmediatamente que esta es una buena definición y que la asignación $C \mapsto UC$ se extiende para definir el funtor "olvido" $U : Cat \to \mathbb{N}$.

   Volviendo a la situación general, decimos que un funtor F es una *equivalencia* si es plenamente fiel y todo objeto de D es isomorfo a un objeto de la forma $F(c)$ con $c \in C$. Por ejemplo, si existe $G : D \mapsto C$ tal que $FG \simeq 1_D$ y $GF \simeq 1_C$, entonces F es una equivalencia y decimos que G es una *inversa homotópica* o *cuasi-inversa* de F. Recíprocamente, si F es una equivalencia entonces es fácil ver que F admite una inversa homotópica.
   Intuitivamente, dos categorías son equivalentes si comparten todas sus características salvo el "tamaño".

   El funtor $F : C_{\mathbb{N}} \to C_{\mathbb{N}}$ toma los objetos de $C_{\mathbb{N}}$, que son conjuntos totalmente ordenados y los asocio a otro conjuntos totalmente ordenado.

   $\square$

2. Dado un objeto fijo a de manera que $b \mapsto a \to b$ y F, una función $F :$ **Hask** $\to$ **Hask**
   Demuestra que F es un funtor.

   Solución:
   Primero para demostrar que F es un funtor definiré las funciones de tal manera que tenagamos un obeto fijo que regrese a su vez un funcion.

   $$f : x \to y$$

$$g : y \to z$$

Que es lo que queremos demostrar, notemos que tenemos que comprobar la composición y la identidad. Con la composición chequemos que $g \circ f \ : x \to z$. Si aplicamos la funcion F tenemos que:

$$F(x) : a \to x$$

$$F(y) : a \to y$$

$$F(z) : a \to z$$

Vemos que $F(f) : F(x) \to F(y)$ es equivalente a $F(f) : (a \to x) \to (a \to y)$, ahora hagamos la composición de $g \circ f$ y le aplicamos la función F y comprovemos que $F(g \circ f) = F(g) \circ F(f)$.

$$\begin{aligned} F(g \circ f) &= F(x \to z) \\ &= F(x) \to F(z) \\ &= (a \to x) \to (a \to z) \end{aligned}$$

Por otro lado.

$$\begin{aligned} F(g) \circ F(f) &= F(y \to z) \circ F(x \to y) \\ &= F(y) \to F(z) \circ F(x) \to (y) \\ &= F(x) \to F(z) \\ &= (a \to x) \to (a \to z) \end{aligned}$$

Por ultimo chequemos la identidad con una función $Id : x \to x$.

$$\begin{aligned} F(Id) &= F(x \to x) \\ &= F(x) \to F(x) \\ &= (a \to x) \to (a \to x) \end{aligned}$$

Es fácil ver que F si es un funtor. □

3. Sean a, b dos objetos fijos. Demuestra que:

   - $F_1(c) = (a, b) \to c$
   - $F_2(c) = a \to (b \to c)$

   Solución1:
   Primero definire una función $f : x \to y$ y $g : y \to z$ y tenemos que

   $$F_1(x) = (a, b) \to x$$

   $$F_1(y) = (a, b) \to y$$

   $$F_1(z) = (a, b) \to z$$

   Vemos que $F(f) : F(x) \to F(y)$ es equivalente a $F(f) : ((a, b) \to x) \to ((a, b) \to y)$, ahora hagamos la composición de $g \circ f$ y le aplicamos la función F y comprovemos que $F(g \circ f) = F(g) \circ F(f)$.

   $$\begin{aligned} F(g \circ f) &= F(x \to z) \\ &= F(x) \to F(z) \\ &= ((a, b) \to x) \to ((a, b) \to z) \end{aligned}$$

Por otro lado.

$$\begin{aligned}
F(g) \circ F(f) &= F(y \to z) \circ F(x \to y)\\
&= F(y) \to F(z) \circ F(x) \to (y)\\
&= F(x) \to F(z)\\
&= ((a,b) \to x) \to ((a,b) \to z)
\end{aligned}$$

Por ultimo chequemos la identidad con una función $Id : x \to x$.

$$\begin{aligned}
F(Id) &= F(x \to x)\\
&= F(x) \to F(x)
\end{aligned}$$

Es fácil ver que F si es un funtor.
<u>*Solución2:*</u>
Primero para demostrar que F es un funtor definiré las funciones de tal manera que tenagamos un obeto fijo que regrese a su vez un funcion.

$$f : x \to y$$

$$g : y \to (z \to w)$$

Que es lo que queremos demostrar, notemos que tenemos que comprobar la composición y la identidad. Con la composición chequemos que $g \circ f \; : x \to (z \to w)$. Si aplicamos la funcion F tenemos que:

$$F(x) : a \to (b \to x)$$

$$F(y) : a \to (b \to y)$$

$$F(z) : a \to (b \to z)$$

$$F(w) : a \to (b \to w)$$

Vemos que $F(f) : F(x) \to F(y)$ es equivalente a $F(f) : (a \to (b \to x)) \to (a \to (b \to y))$, ahora hagamos la composición de $g \circ f$ y le aplicamos la función F y comprovemos que $F(g \circ f) = F(g) \circ F(f)$.

$$\begin{aligned}
F(g \circ f) &= F(x \to (z \to w))\\
&= F(x) \to F(z \to w)\\
&= (a \to (b \to x)) \to (F(z) \to F(w))\\
&= (a \to (b \to x)) \to ((a \to (b \to z)) \to (a \to (b \to w)))
\end{aligned}$$

Por otro lado.

$$\begin{aligned}
F(g) \circ F(f) &= F(y \to (z \to w)) \circ F(x \to y)\\
&= (F(y) \to F(z \to w)) \circ (F(x) \to F(y))\\
&= (F(y) \to (F(z) \to F(w)) \circ (F(x) \to F(y))\\
&= ((a \to (b \to y)) \to ((a \to (b \to z) \to (a \to (b \to w)) \circ ((a \to (b \to x)) \to (a \to (b \to y)))\\
&= (a \to (b \to x)) \to ((a \to (b \to z)) \to (a \to (b \to w)))
\end{aligned}$$

Por ultimo chequemos la identidad con una función $Id : x \to x$.

$$\begin{aligned} F(Id) &= F(x \to x) \\ &= F(x) \to F(x) \\ &= (a \to (b \to x)) \to (a \to (b \to x)) \end{aligned}$$

Es fácil ver que F si es un funtor. $\square$

4. Demuestra que existen $\eta_1 : F_1 \Rightarrow F_2$ y $\eta_2 : F_2 \Rightarrow 1$ transformaciones naturales tales que $\eta_2 \circ \eta_1 = F_1$ y $\eta_1 \circ \eta_2 = F_2$.

   <u>*Solución:*</u>
   Para demostrar que existe basta con dar un diagrama que lo cumple de modo que la composición dada se cumpla. Para este caso en particular usaremos el diagrama visto en clase y lo analizaremos.

$$\begin{tikzcd} F_1(a) \arrow[r, "F_1(f)"] \arrow[d, leftrightarrow, "\eta_{2a}"', "\eta_a"] & F_1(b) \arrow[d, leftrightarrow, "\eta_{2b}"', "\eta_b"] \\ F_2(a) \arrow[r, "F_2(f)"'] & F_2(b) \end{tikzcd}$$

   Podemos ver que la composición se da, y temenos las η de manera que cumple las composiciones de manera que todas son la identidad, ya que tenemos que el final F_2 al hacer la composición, conpone a la identidad. $\square$

5. Sea $\eta : [\,] \Rightarrow [\,]$ como sigue:

   - $\eta_a : [a] \to [a]$
   - $xs \mapsto sort\ xs$

   donde *sort* es cualquier función que ordena una lista dada. ¿Es η una transformación natural? Justifica tu respuesta.

   <u>*Solución:*</u>
   No es una Transformacion natural, chequemos este contra-ejemplo.
   Supongamos que tenemos la funcion $f : Int \to Boolean$ de manera que cumple que.

$$f = \begin{cases} True & si\ x\ es\ par \\ False & si\ x\ es\ impar \end{cases}$$

Si nos damos cuenta, para que sea una transformación natural es importante que cumpla que:

$$(map\ f)(sort\ l) = sort(map\ f\ l)$$

Pero si vemos con el ejemplo en el que $l = [2, 1, 3]$ tenemos lo siguiente.

$$\begin{aligned}(map\ f)(sort\ [2,1,3]) &= sort(map\ f\ [2,1,3])\\ (map\ f)[1,2,3] &= sort[True, False, False]\\ [False, True, False]\ &!= Error\end{aligned}$$

Como podemos ver, no puede que sea una transformacion natural.

$\square$

6. Demuestra que $\eta : Maybe \Rightarrow [\ ]$ definida como:

   - $\eta_a : Maybe\ a \to [a]$
   - $Nothing \mapsto [\ ]$
   - $Just\ x \mapsto [x]$

   es una transformación natural.

   Solución:
   Si $F, G\ :\ C \to D$ son dos funtores que van de la categoría C a la categoría D, una transformación natural $\eta : F \Rightarrow G$ consiste en lo siguiente:

   - Por cada objeto x en $Ob(C)$, un morfismo $\eta x : F(x) \to G(x)$ tal que, si $a \xrightarrow{f} b$ es cualquier morfismo en C, entonces:

     $$G(f) \circ \eta_a = \eta_b \circ F(f)$$

   $$(map\ f)(Nothing) = (map\ f)[\ ] = [\ ] = (Nothing)(\_)$$

   $$(map\ f)(Just\ x) = (map\ f)[x] = f[x] = (map\ f)[x] = (Just\ fx)$$

   Si nos damos la composición la cumple y podemos concluir que η es una transformación natural. $\square$

7. Demuestra que **Maybe** : **Hask** $\to$ **Hask** es un funtor. Además:

   - Demuestra que **Maybe** es una monada (desde el punto de vista categorico).
   - Demuestra que **Maybe** es una terna de Kleisli.

Solución 1:

Para probar que **Maybe** es un funtor, notemos que Haskell define una clase de tipo **Functor** que contiene una función *fmap* que para lograr esta cumpla que un data sea un funtor, las instancias de Functor deben obedecer las dos leyes siguientes donde:

```
fmap id == id
fmap (f . g) == (fmap f . fmap g)
```

Un valor de **Maybe** a es *Just a* o *Nothing*; la instancia debería permitirnos aplicar una función en a en *Just a* o fallar silenciosamente en *Nothing* sin cambiar nada más. Por lo tanto, *fmap* se define como lo siguiente.

```
instance  Functor Maybe  where
    fmap _ Nothing        = Nothing
    fmap f (Just a)       = Just (f a)
```

Primera Propiedad.
Probaré por casos sobre una m de **Maybe** a, de manera que *fmap id* $m ==$ *id* m.
Caso 1. $m == Nothing$.

$$\begin{aligned} fmap\ id\ m &== fmap\ id\ Nothing \\ &== Nothing \\ &== id\ m \end{aligned}$$

Caso 2. $m == Just\ a$.

$$\begin{aligned} fmap\ id\ m &== fmap\ id\ (Just\ a) \\ &== Just\ (id\ a) \\ &== Just\ a \\ &== m \\ &== id\ m \end{aligned}$$

Segunda Propiedad.
Probaré por casos sobre una m de **Maybe** a, de manera que *fmap* $(f\ \cdot\ g)\ m == (fmap\ f\ \cdot\ fmap\ g)\ m$.
Caso 1. $m == Nothing$.

$$\begin{aligned} fmap\ (f\ \cdot\ g)\ m &== fmap\ (f\ \cdot\ g)\ Nothing \\ &== Nothing \\ (fmap\ f\ \cdot\ fmap\ g)\ m &== fmap\ f\ (fmap\ g\ Nothing) \\ &== fmap\ f\ Nothing \\ &== Nothing \end{aligned}$$

Caso 2. $m == Just\ a$.

$$\begin{aligned} fmap\ (f\ \cdot\ g)\ m &== fmap\ (f\ \cdot\ g)\ (Just\ a) \\ &== Just\ ((f\ \cdot\ g)\ a) \\ (fmap\ f\ \cdot\ fmap\ g)\ m &== fmap\ f\ (fmap\ g\ (Just\ a)) \\ &== fmap\ f\ (Just\ (g\ a)) \\ &== Just\ (f\ (g\ a)) \\ &== Just\ ((f\ \cdot\ g)\ a) \end{aligned}$$

□

<u>*Solución 2:*</u>

Para probar que **Maybe** es una monada desde el punto de vista categorico, tenemos que cumplir las siguientes condiciones:

- T : **Hask** $\rightarrow$ **Hask** un funtor.
- $\eta : Id_{\mathbf{Hask}} \Rightarrow T$ una transformación natural.
- $\mu : T \circ T \Rightarrow T$ una transformación natural.

El funtor que vamos a dar sería el que definimos previamente, así que el primer inciso podemos darlo por obvio.

Para los siguientes incisos debemos dar una *eta* y una *mu* que sean una transformación natural, veamos la siguiente definición en Haskell.

Definamos a *eta* como:

$$\eta :: a \rightarrow Maybe\ a$$

Veamos que si es una transformación natural valida, ya que con sea f una función que se aplica con *map* de manera que:

$$(map\ f)\ ([a]) \rightarrow g$$

y a su ves que si f se le aplica a g, nos de como resultado Maybe a. Como meterlo a la cajita usando el funtor y regresando $Justa$, en el caso de Nothing no es necesario definirlo ya que no regresa absolutamente nada.

Ahora la μ veamos que debe ser una composición, y definamos la siguiente μ.

$$\mu\ ::\ Maybe\ a \rightarrow (a \rightarrow Maybe\ b) \rightarrow Maybe\ b$$

Es fácil ver que tenemos la composicion de dos transformaciones que regresa una transformacion, en este caso, tenemos dos casos, cuando es $Justx$ podemos ver que la composición es solamente aplicarle la función a lo que esta adentro de la çajita", algo como $\mu\ m\ f$. Si m es $Just\ x$ sería solo componer aplicando la función a "x", pero si es "Nothing"no importa que funcion sea, el resultado seguirá siendo nada. □

<u>*Solución 3:*</u>

Si $(T, \eta, *)$ es una terna de Kleisli sobre la categoría C, la categoría de Kleisli denotada C_T se define como:

- $Ob(C_T) = Ob(C)$
- $Hom_{C_T}(a,b) = Hom_C(a, T(b))$
- $id_a = \eta_a$
- Para f en $Hom_{C_T}(a,b)$, g en $Hom_{C_T}(b,c)$, definimos $g \circ f$ como $g^* \circ f : a \to T(c)$.

Esto se reduce a probar que:

- asociatividad: $h^* \circ (g^* \circ f) = (h^* \circ g)^* \circ f$
- identidad izquierda: $\eta_b^* \circ f = f$
- identidad derecha: $f \circ \eta_a^* = f$

Podemos pensar en $Maybe$ como: $\eta_a = Just$; $f : a \to b$, $f^*(Nothing) = Nothing$ y $f^*(Just\ a) = f(a)$

Probemos la asociatividad:
Caso 1. Para Nothing.

$$h^*(g^*(Nothing)) = (h^*(Nothing) = Nothing$$

y por otro lado

$$(h^* \circ g)^* \circ (Nothing) = Nothing$$

Caso 2. Para Just a.

$$h^*(g^*(Just\ a)) = (h^*(ga) = h(g(a))$$

y por otro lado

$$(h^* \circ g)^* \circ (Justa) = (h^*)g(Just\ a) = (h^*)g(a) = h(g(a))$$

Para la identidad izquierda:

$$Just^* \circ m = Just\ m = m$$

y de manera analoga la identidad derecha. □