FUNDAMENTOS DE COMPUTACIÓN
Repartido: Otros Tipos Finitos

Así como definimos a los Booleanos, podemos repetir el procedimiento y definir nuevos tipos, cuyos elementos serán introducidos mediante la enumeración explícita de los mismos. En Haskell los mismos reciben también el nombre de **Tipos Enumerados.**

a) Los Puntos Cardinales
Un ejemplo clásico en cursos iniciales de Programación es el tipo finito que contiene los puntos cardinales:

$$\frac{}{\textbf{PCard tipo}}$$

y sus valores son los que representan esos objetos:

$$\frac{}{\textbf{Norte :: PCard}} \qquad \frac{}{\textbf{Este :: PCard}} \qquad \frac{}{\textbf{Sur :: PCard}} \qquad \frac{}{\textbf{Oeste :: PCard}}$$

En Haskell se declara el tipo del siguiente modo:

data PCard where { Norte :: PCard , Este :: PCard , Sur :: PCard , Oeste :: PCard }

Del mismo modo que con Bool, asociado a este tipo tendremos su expresión case, que nos permite realizar análisis de casos:

$$\frac{\textbf{e :: PCard} \qquad \textbf{e1 :: t} \qquad \textbf{e2 :: t} \qquad \textbf{e3 :: t} \qquad \textbf{e4 :: t}}{\textbf{case e of \{ Norte} \rightarrow \textbf{e1 ; Este} \rightarrow \textbf{e2 ; Sur} \rightarrow \textbf{e3 ; Oeste} \rightarrow \textbf{e4 \} :: t}}$$

y también las siguientes igualdades, cuyas partes izquierdas serán redexes al momento de computar:

case Norte of { Norte → e1 ; Este → e2 ; Sur → e3 ; Oeste → e4 } = e1
case Este of { Norte → e1 ; Este → e2 ; Sur → e3 ; Oeste → e4 } = e2
case Sur of { Norte → e1 ; Este → e2 ; Sur → e3 ; Oeste → e4 } = e3
case Oeste of { Norte → e1 ; Este → e2 ; Sur → e3 ; Oeste → e4 } = e4

Utilizando las expresiones case podremos definir funciones sobre el tipo PCard, por ejemplo **siguiente :: PCard → PCard**, que calcula el siguiente de un punto cardinal (en sentido horario):

siguiente = λp→ case p of { Norte → Este ;
Este → Sur ;
Sur → Oeste ;
Oeste → Norte }

También se puede definir la función **opuesto:: PCard → PCard**, que calcula el opuesto de un punto cardinal:

opuesto = λp→ case p of { Norte → Sur;
Este → Oeste;
Sur → Norte;
Oeste → Este }

Y finalmente, también tendremos asociado un método de demostración de propiedades de los puntos cardinales, la demostración por casos:
Para demostrar una afirmación de la forma **(∀p :: PCard) P(p)**, *alcanza con demostrar:*

- **el caso Norte:** o sea, demostrar **P(Norte)**
- **el caso Este:** o sea, demostrar **P(Este)**
- **el caso Sur:** o sea, demostrar **P(Sur)**
- **el caso Oeste:** o sea, demostrar **P(Oeste).**

Como ejemplo, mostramos la demostración de una propiedad que cumplen las funciones anteriormente definidas:

Lema1$_{\mathbf{PCard}}$: (∀p :: PCard) siguiente (opuesto p) = opuesto (siguiente p)
Dem. Por casos en p :: PCard.
Caso Norte: siguiente (opuesto Norte) $=^?$ opuesto (siguiente Norte)

Dem.

siguiente (opuesto Norte)	opuesto (siguiente Norte)
= (def. opuesto, β, case)	= (def. siguiente, β, case)
siguiente Sur	opuesto Este
= (def. siguiente, β, case)	= (def. opuesto, β, case)
Oeste	Oeste

Ambas expresiones son iguales por reducir a la misma expresión

Caso Este: siguiente (opuesto Este) $=^?$ opuesto (siguiente Este)

Dem.

siguiente (opuesto Este)	opuesto (siguiente Este)
= (def. opuesto, β, case)	= (def. siguiente, β, case)
siguiente Oeste	opuesto Sur
= (def. siguiente, β, case)	= (def. opuesto, β, case)
Norte	Norte

Ambas expresiones son iguales por reducir a la misma expresión

Caso Sur: siguiente (opuesto Sur) $=^?$ opuesto (siguiente Sur)

Dem.

siguiente (opuesto Sur)	opuesto (siguiente Sur)
= (def. opuesto, β, case)	= (def. siguiente, β, case)
siguiente Norte	opuesto Oeste
= (def. siguiente, β, case)	= (def. opuesto, β, case)
Este	Este

Ambas expresiones son iguales por reducir a la misma expresión

Caso Oeste: siguiente (opuesto Oeste) $=^?$ opuesto (siguiente Oeste)

Dem.

siguiente (opuesto Oeste)	opuesto (siguiente Oeste)
= (def. opuesto, β, case)	= (def. siguiente, β, case)
siguiente Este	opuesto Norte
= (def. siguiente, β, case)	= (def. opuesto, β, case)
Sur	Sur

Ambas expresiones son iguales por reducir a la misma expresión

b) Los Días de la Semana
Otro ejemplo clásico de tipo enumerado es el de los días de la semana, definido por las siguientes reglas:

$$\frac{}{\textbf{Dia tipo}}$$

$$\frac{}{\textbf{Lu :: Dia}} \quad \frac{}{\textbf{Ma :: Día}} \quad \frac{}{\textbf{Mi :: Día}} \quad \frac{}{\textbf{Ju :: Día}} \quad \frac{}{\textbf{Vl :: Día}} \quad \frac{}{\textbf{Sa :: Día}} \quad \frac{}{\textbf{Do :: Día}}$$

En Haskell se declara el tipo del siguiente modo:

data Día where {Lu :: Dia , Ma :: Día , Mi :: Día , Ju :: Día , Vl :: Día , Sa :: Día , Do :: Día}

La expresión **case** para este tipo será la siguiente:

$$\frac{\textbf{e :: Día} \quad \textbf{e1 :: t} \quad \textbf{e2 :: t} \quad \textbf{e3 :: t} \quad \textbf{e4 :: t} \quad \textbf{e5 :: t} \quad \textbf{e6 :: t} \quad \textbf{e7 :: t}}{\textbf{case e of \{Lu} \rightarrow \textbf{e1; Ma} \rightarrow \textbf{e2; Mi} \rightarrow \textbf{e3; Ju} \rightarrow \textbf{e4; Vi} \rightarrow \textbf{e5; Sa} \rightarrow \textbf{e6; Do} \rightarrow \textbf{e7\} :: t}}$$

con las siete igualdades correspondientes:
case Lu of {Lu → e1; Ma → e2; Mi → e3; Ju → e4; Vi → e5; Sa → e6; Do → e7} = e1
case Ma of {Lu → e1; Ma → e2; Mi → e3; Ju → e4; Vi → e5; Sa → e6; Do → e7} = e2
.
.
.
case Do of {Lu → e1; Ma → e2; Mi → e3; Ju → e4; Vi → e5; Sa → e6; Do → e7} = e7

Podemos definir funciones sobre el tipo Día, por ejemplo:

```
laborable :: Día → Bool
laborable = λd→ case d of { Sa → False ;
                            Do → False ;
                             _   → True }
```

Observar el uso del "_", llamado comodín o *"wild card".*
Haskell evalúa desde arriba hacia abajo los casos posibles para d. El comodín se usa para abarcar todos los casos posibles que quedan debajo de los casos concretos (Sa y Do en este caso).

También habrá un método de demostración por casos asociado al tipo Día.
Para demostrar que **(∀d :: Día) P(d) se cumple***, alcanza con demostrar:*

- **el caso Lu:** o sea, demostrar **P(Lu)**
- **el caso Ma:** o sea, demostrar **P(Ma)**
  .
  .
- **el caso Do:** o sea, demostrar **P(Do).**