Polimorfismo

Repasando...

- Se llama polimorfismo a una función que puede aplicarse a distintos tipos de datos (sin redefinirla).
- se usa cuando el comportamiento de la función no depende paramétricamente del tipo de sus argumentos
- lo vimos en el lenguaje de especificación con las funciones genéricas.
- En Haskell los polimorfismos se escriben usando variables de tipo y conviven con el tipado fuerte.
- Ejemplo de una función polimórfica: la función identidad.

Variables de tipos

¿Qué tipo tienen las siguientes funciones?

```
identidad x = x

primero x y = x

segundo x y = y

constante5 x y z = 5
```

Variables de tipo

- Son parámetros que se escriben en la signatura usando variables minúsculas
- En lugar de valores, denotan tipos
- Cuando se invoca la función se usa como argumento el tipo del valor

Variables de tipo (cont.)

Funciones con variables de tipo

```
identidad :: t -> t
identidad x = x

primero :: tx -> ty -> tx
primero x y = x

segundo :: tx -> ty -> ty
segundo x y = y

constante5 :: tx -> ty -> tz -> Int
constante5 x y z = 5

mismoTipo :: t -> t -> Bool
mismoTipo x y = True
```

Si dos argumentos deben tener el mismo tipo, se debe usar la misma variable de tipo

- Luego, `primero True 5 :: Bool`, pero `mismoTipo 1 True 0` no tipa

Especificación de un problema: Extensión

Variables de tipo

- Vamos a querer describir funciones polimórficas con nuestro lenguaje de especificación
- Veamos cómo podemos hacerlo...

Especificación de un problema: Extensión

Variables de tipo

problema *nombre*(*parámetros*) : tipo de dato del resultado {
  requiere *etiqueta*: { condiciones sobre los parámetros de entrada }
  asegura *etiqueta*: { condiciones sobre los parámetros de salida }
}

- *nombre*: nombre que le damos al problema
  - será resuelto por una función con ese mismo nombre
- *parámetros*: lista de parámetros separada por comas, donde cada parámetro contiene:
  - Nombre del parámetro
  - Tipo de datos del parámetro o una **variable de tipo**
- *tipo de dato del resultado*: tipo de dato del resultado del problema (inicialmente especificaremos funciones) o una **variable de tipo**
  - En los asegura, podremos referenciar el valor devuelto con el nombre de **res**
- *etiquetas*: son nombres opcionales que nos servirán para nombrar declarativamente a las condiciones de los requiere o aseguras.

Especificación de un problema: Extensión

Variables de tipo

- El símbolo o nombre (letra) de la variable de tipo no se corresponde con ninguno de los tipos de datos conocidos. Es una representación genérica.
- Cada ocurrencia de una variable de tipo, siempre representa al mismo tipo de datos.

problema $segundo(x : U, y : T) : T\{$
  asegura devuelveElSegundo: $\{res = y\}$
$\}$

problema $cantidadDeApariciones(s : seq\langle T\rangle, e : T) : \mathbb{Z}\ \{$
  asegura: $\{res = \sum_{i=0}^{|s|-1}(\text{if } s[i] = e \text{ then } 1 \text{ else } 0 \text{ fi})\}$
$\}$

Especificación de un problema: Extensión

Variables de tipo con restricciones

- Se puede restringir los posibles tipos de una variable de tipo mediante un requiere

problema $suma(x : T, y : T) : T\{$
  requiere: $\{T \in [\mathbb{N}, \mathbb{Z}, \mathbb{R}]\}$
  asegura: $\{res = x + y\}$
$\}$