PROGRAMACIÓN FUNCIONAL

Recursión de Cola y Teoremas de Dualidad

Recursión de cola

- Recursión de cola: foldl
- Teoremas de dualidad
- Técnica de generalización

Recursión de Cola

- En la función foldr, el llamado recursivo es argumento de otras funciones.
- ¿Podrá cambiarse la definición para que el llamado recursivo sea la función principal?
- ¿Por qué sería interesante?
  - Con orden normal de evaluación, la recursión sería la que determina qué se computa
  - Puede mejorar la eficiencia del programa

Recursión de Cola

- ¿Cómo quedaría esta definición?

```
foldl :: ??
foldl f a [ ] = a
foldl f a (x:xs) = foldl f (a `f` x) xs
```

- Ejemplo:

```
foldl (+) 0 [ 1, 2, 3 ]                    →
  foldl (+) (0+1) [ 2, 3 ]                 →
  foldl (+) ((0+1)+2) [ 3 ]                →
  foldl (+) (((0+1)+2)+3) [ ]              →
  (((0+1)+2)+3)
```

Recursión de Cola

- El esquema foldl
  - ¿es equivalente al de foldr?
  - ¿en qué casos?
- Veamos

  foldl f a [x1, x2, .. , xn] →
  (...((a `f` x1) `f` x2) `f` ... `f` xn)

  mientras que

  foldr f a [x1, x2, .. , xn] →
  (x1 `f` (x2 `f` ... `f` (xn `f` a)...))

Teoremas de Dualidad

- Primer Teorema de Dualidad:

> Cuando f es asociativa y a es su neutro, se cumple que, para toda lista finita xs,
>
> foldr f a xs = foldl f a xs

- ¿Cómo lo demostramos?
  - Podríamos usar inducción sobre xs

  o bien
  - podríamos demostrar un teorema más general, y obtener éste como caso particular

Teoremas de Dualidad

- Consideremos las siguientes definiciones:

  reverse = foldr postfix [ ]
  postfix x xs = xs ++ [x]

  rev = foldl prefix [ ]
  prefix xs x = [x] ++ xs

- ¿Es cierto que reverse = rev?
- ¿Cómo lo demostramos? ¿Sirve el 1er Teorema de Dualidad?

Teoremas de Dualidad

- Segundo Teorema de Dualidad:

Sean f, g y a cumpliendo que para todos x, y, z

```
x `f` (y `g` z) = (x `f` y) `g` z
x `f` a = a `g` x
```

entonces, para toda lista finita xs

```
foldr f a xs = foldl g a xs
```

- ¿Cómo lo demostramos?
  - Por inducción en la estructura de xs
  - Requiere un resultado auxiliar (lema)

Teoremas de Dualidad

- El 1er Teorema de Dualidad es consecuencia del 2do, tomando f = g
- La igualdad reverse=rev también es consecuencia del 2do Teorema de Dualidad, tomando f = postfix, g = prefix y a = [ ]
  - Verificar que

    x \`postfix\` (xs \`prefix\` y) = (x \`postfix\` xs) \`prefix\` y

    y que

    postfix x [ ] = prefix [ ] x

Teoremas de Dualidad

- Tercer Teorema de Dualidad:

> Para toda lista finita xs, vale que
>
> foldr f a xs = foldl (flip f) a (reverse xs)
>
> siendo
>
> flip f y x = f x y

- ¿Cómo lo demostramos?
  - Inducción en la estructura de xs
  - Usar el siguiente lema: para toda lista finita xs y para todo x, se cumple que para todo a,

    foldl f a (xs++[x]) = f (foldl f a xs) x

Generalización

- ¿Qué sucede si tratamos de mostrar lo siguiente?

Lema: para toda lista finita xs y para todo x, se cumple que

foldl f a (xs++[x]) = f (foldl f a xs) x

- **Demostración**: por inducción en la estructura de xs
  - <u>Caso base</u>: xs = [ ]
    - (++.1), (foldl.2), (foldl.1), (foldl.1)
  - <u>Caso inductivo</u>: xs = (y:ys)
    HI: foldl f a (ys++[x]) = f (foldl f a ys) x
    TI: foldl f a ((y:ys)++[x]) = f (foldl f a (y:ys)) x
    - (++.2), (foldl.2), ¿HI?...

Generalización

- ¿Qué falla? ¡La HI predica sobre un único a!
  - Debemos generalizar y predicar sobre *cualquier* a

Lema: para toda lista finita xs y para todo x, se cumple que , *cualquiera sea* *a*, foldl f a (xs++[x]) = f (foldl f a xs) x

- **Demostración**: por inducción en la estructura de xs
  - Caso base: ídem
  - Caso inductivo: xs = (y:ys)

    HI: ∀a. foldl f a (ys++[x]) = f (foldl f a ys) x

    TI: ∀a. foldl f a ((y:ys)++[x]) = f (foldl f a (y:ys)) x

    - (++.2), (foldl.2), (HI, con a'=f a y), (foldl.2)

Teoremas de Dualidad

- Demostración del 2do Teorema de Dualidad
  - por inducción sobre la estructura de la lista, utilizando el lema que se enuncia a continuación
- **Lema:**

  para todo x y toda lista finita xs, y siendo f y g como en el teorema, se cumple que para todo a,

  x \`f\` (foldl g a xs) = foldl g (x \`f\` a) xs
- **Demostración**: por inducción sobre xs
- ¡Observar que el lema ya está generalizado!

Resumen

- Recursión de cola (foldl)
- Teoremas de dualidad
- Generalización