Recursividad sobre estructuras lineales

Estructuras de Datos

Grado en Ingeniería Informática, del Software y Computadores

Universidad de Málaga

Recursividad sobre listas: cabeza y cola

Esquema de función recursiva:

```
procesa [] = solBase
procesa (x:xs) =
        f x (procesa xs)
```

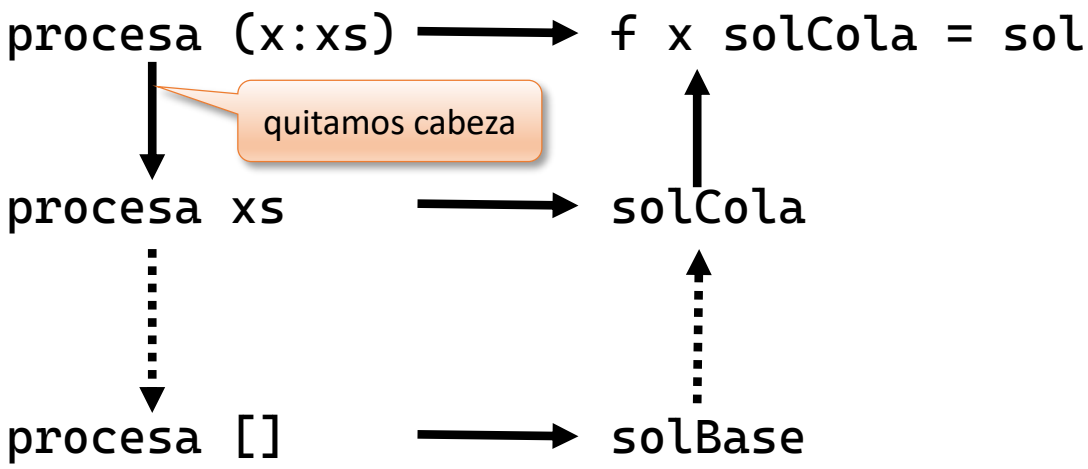
cabeza

solución cola

Ejemplo de función recursiva:

```
suma :: [Int] -> Int
suma [] = 0
suma (x:xs) =
       x + (suma xs)
```

Proceso recursivo:



Recursividad sobre pilas: cima y resto

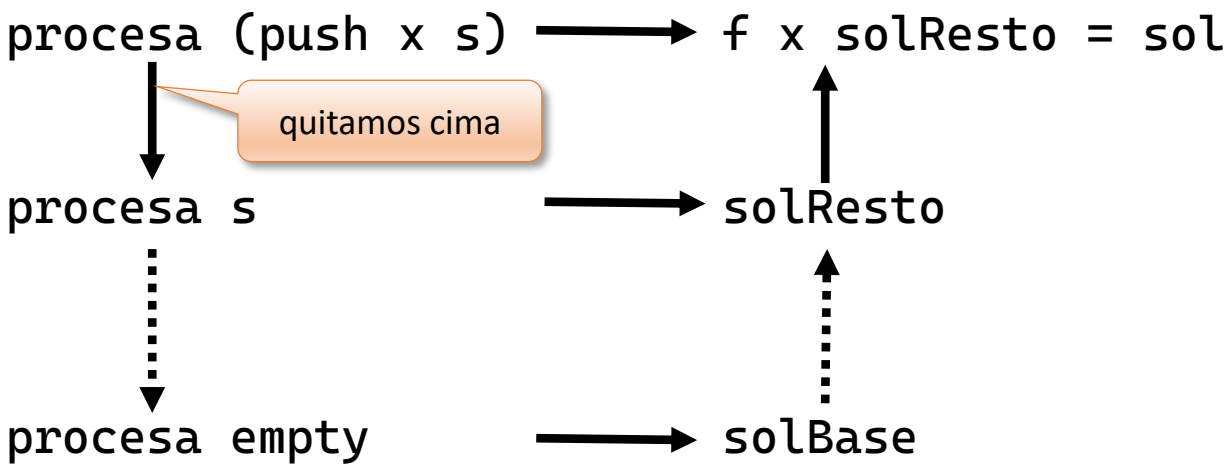
Esquema de función recursiva:

```
procesa s
  | isEmpty s = solBase
  | otherwise =
      f (top s) (procesa (pop s))
```

cima

solución resto

Proceso recursivo:



Ejemplo de función recursiva:

```
aLista :: Stack a -> [a]
aLista s
  | isEmpty s = []
  | otherwise =
      top s : aLista (pop s)
```

Recursividad sobre colas: primero y resto

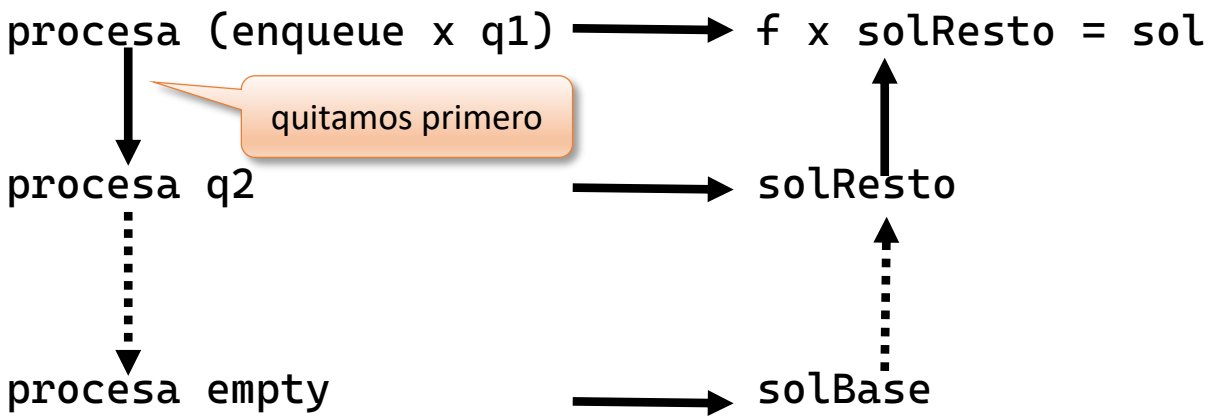
Esquema de función recursiva:

```
procesa q
 | isEmpty q = solBase
 | otherwise =
    f (first s) (procesa (dequeue q))
```

primero

solución resto

Proceso recursivo:



Ejemplo de función recursiva:

```
aLista :: Queue a -> [a]
aLista q
 | isEmpty q = []
 | otherwise =
      first q : aLista (dequeue q)
```

q2 = dequeue (enqueue x q1)

Recursividad sobre conjuntos: alguno y resto

Esquema de función recursiva:

```
procesa s
  | isEmpty s = solBase
  | otherwise =
      f (??? s) (procesa (??? s))
```

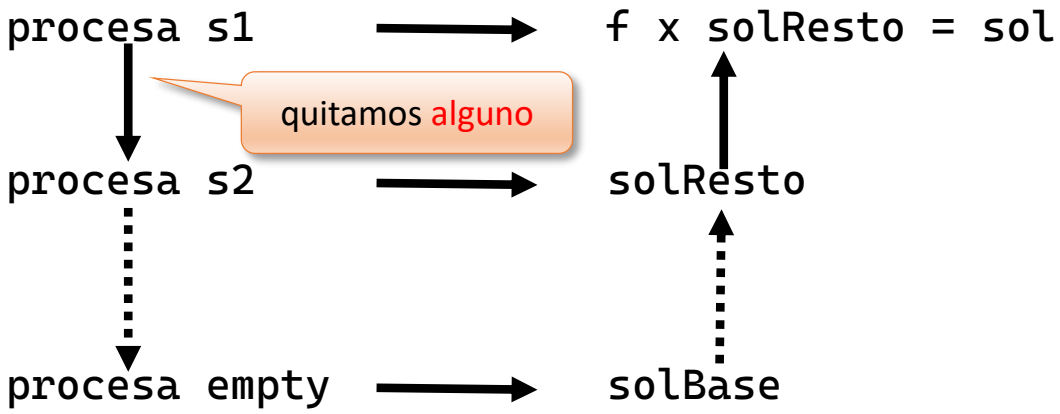
algún elemento

solución resto

Ejemplo de función recursiva:

```
aLista :: Set a -> [a]
aLista s
  |isEmpty s = []
  |otherwise =
      ??? s : aLista (??? s)
```

Proceso recursivo:



s2 = s1 \ {x} donde x pertenece a s1

Problema:

- no tenemos una función para quitar algún elemento
- `delete x s` no vale; menciona elemento a quitar
- listas, pilas y colas tienen una posición distinguida
- en los conjuntos no existe una posición distinguida

Plegado a la derecha de listas: cabeza y cola

Función de plegado:

```
foldr :: (a->a->b) -> b -> [a] -> b
foldr f solBase [] = solBase
foldr f solBase (x:xs) =
    f x (foldr f solBase xs)
```

cabeza

solución cola

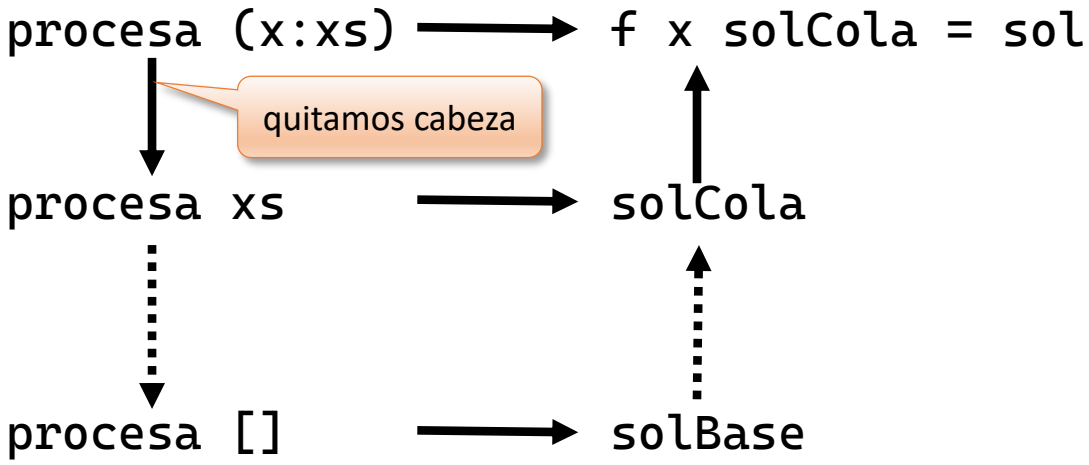
Ejemplo de plegado:

```
suma :: [Int] -> Int
suma xs = foldr (+) 0 xs
```

f

solBase

Proceso recursivo:



Plegado de conjuntos: alguno y resto

Tipo de la función de plegado:

```
fold :: (a->b->b) -> b -> Set a -> b
```

algún elemento

solución resto

solución conjunto vacío

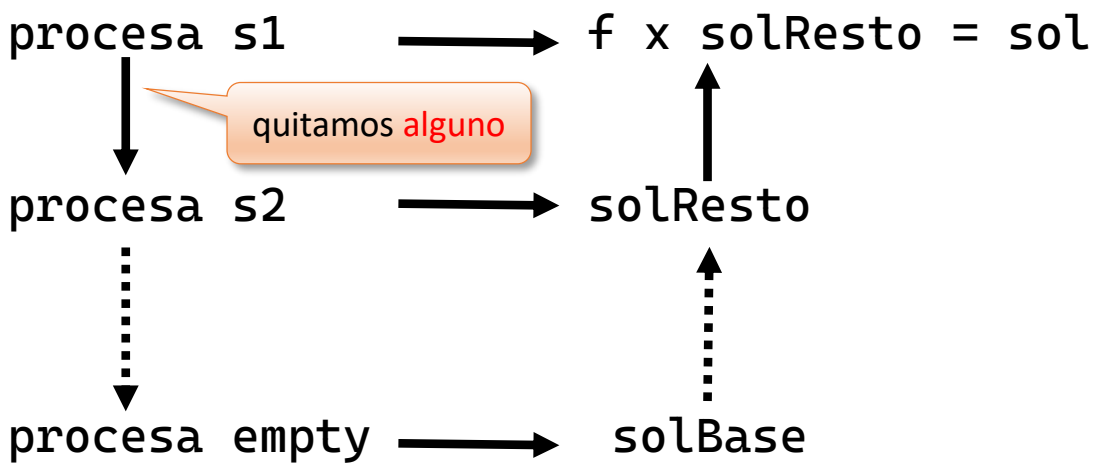
Ejemplo de plegado:

```
suma :: Set Int -> Int
suma s = fold (+) 0 s
```

f

solBase

Proceso recursivo:



s2 = s1 \ {x} donde x pertenece a s1

Solución:

- función de plegado de conjuntos
- visita todos los elementos en algún orden
- el código cliente no conoce el orden de visita

¿Por qué los conjuntos son distintos?

- Listas, pilas y colas son estructuras **lineales**:

  x1, x2, x3, x4, …, xn
- Todo elemento xi tiene un **predecesor** (excepto el primero)
- Todo elemento xi tiene un **sucesor** (excepto el último)
- Existe una posición **distinguida** (cabeza, cima, primero)
- Los conjuntos **no** son una estructura lineal
- Nuestra **representación física** de conjuntos es lineal
- Veremos otras representaciones no lineales más **eficientes**