

# Manejo de errores para $\mathcal{T}$

**José de Jesús Lavalle Martínez**

<http://aleteya.cs.buap.mx/~jllavalle/>

Benemérita Universidad Autónoma de Puebla  
Facultad de Ciencias de la Computación  
Licenciatura en Ciencias de la Computación  
Fundamentos de Lenguajes de Programación  
CCOS 255

Otoño 2020

- 1 Introducción
- 2 Definiciones mutuamente recursivas
- 3 Ejercicios

# Introducción

```
- sc t;  
> val it = sc t : T  
- evalast(sc t);  
> val it = sc t : T
```

```
- ifte(sc z, t, f);  
> val it = ifte(sc z, t, f) : T  
- evalast(ifte(sc z, t, f));  
! Uncaught exception:  
! Match  
- evalast(sc z);  
> val it = sc z : T  
-
```

- La idea es que un tipo  $x$  se define recursivamente pero no en términos de si mismo, sino de otro tipo  $y$ , a su vez el tipo  $y$  está definido en términos del tipo  $x$ .

- Lo mismo puede ocurrir con la definición de una función.

- Por supuesto en ML si un tipo está definido mediante recursión mutua cualquier función sobre ese tipo también debe estar definido mediante recursión mutua.

## Ejemplo de una función mutuamente recursiva

```
fun even 0 = true
|   even n = odd  (n-1)
and odd 0 = false
|   odd  n = even (n-1);
```

```
- even 4;
> val it = true : bool
- even 3;
> val it = false : bool
- odd 4;
> val it = false : bool
- odd 3;
> val it = true : bool
-
```



# Ejemplo de un tipo y una función mutuamente recursivas I

```
datatype 'a tree = Empty | Node of 'a * 'a forest  
and 'a forest = Nil | Cons of 'a tree * 'a forest;
```

# Ejemplo de un tipo y una función mutuamente recursivas I

```
datatype 'a tree = Empty | Node of 'a * 'a forest
and 'a forest = Nil | Cons of 'a tree * 'a forest;

fun sizeTree Empty = 0
    | sizeTree(Node(_, f)) = 1 + sizeForest f
and sizeForest Nil = 0
    | sizeForest(Cons(t, f')) = sizeTree t + sizeForest f';
```

## Ejemplo de un tipo y una función mutuamente recursivas II

```
- Node(1, Cons(Node(2, Nil), Nil));  
> val it = Node(1, Cons(Node(2, Nil), Nil)) : int tree  
- size_tree(Node(1, Cons(Node(2, Nil), Nil)));  
> val it = 2 : int
```

# Ejemplo de un tipo y una función mutuamente recursivas II

```
- Cons(Node(2, Nil),Nil);  
> val it = Cons(Node(2, Nil), Nil) : int forest  
- size_forest(Cons(Node(2, Nil),Nil));  
> val it = 1 : int  
-
```

- El libro de texto *Types and Programming Languages* en el ejercicio 3.5.16 que está en la página 42, propone aumentar las categorías sintácticas `badnat`, `badbool` y cuatro reglas para la evaluación en un paso que permitan en conjunto controlar los errores en tiempo de ejecución.

- Pero nosotros queremos evitar los errores en tiempo de ejecución desde la parte sintáctica, aprovechando las características de ML.

# Separación de tipos en ML para $\mathcal{T}$

```
datatype B = t | f | isz of N | ifteb of B * B * B  
and      N = z | sc of N | pd of N | iften of B * N * N;  
datatype T = b of B | n of N;
```

- 1 Construya tres árboles y tres bosques.
- 2 Aplique las funciones `sizeTree` y `sizeForest` a lo que construyó en el ejercicio anterior.
- 3 Construya las funciones:
  - 1 `evalb = fn : B -> B.`
  - 2 `evaln = fn : N -> N`
  - 3 `evalast = fn : T -> T`
- 4 Pruebe `evalast` con el archivo `casosPruebaManejoDeErrores.ml` que encontrará en mi página.