

Grupo	Grupo Haskell, Curry & Minions
Campanello, José Luis	1207/88
Delgado, Gonzalo Sebastián	38/17
Rankov González, Jorge Augusto Germán	714/23
Santos, Diego Hernán	874/03

# Resolución Ejercicio 12 TP 1

Más abajo se puede observar el enunciado completo.

## Resolución Ejercicio 12

### A – Definir el predicado unario

Definir el predicado unario correspondiente a una demostración por inducción estructural (¿en qué estructura?) de esta propiedad.

Dada la propiedad:

$$\forall e :: \text{Expr} \cdot \text{cantLit } e = S (\text{cantOp } e)$$

El predicado unario es:

$$P(e) = \text{cantLit } e = S (\text{cantOp } e)$$

La inducción estructural se realizará sobre la estructura Expr, definida en el Módulo Expr del TP como:

```
data Expr = Const Float
          | Rango Float Float
          | Suma Expr Expr
          | Resta Expr Expr
          | Mult Expr Expr
          | Div Expr Expr
  deriving (Show, Eq)
```

## B – Esquema formal de Inducción Estructural

Definir el esquema formal de inducción estructural correspondiente a dicha demostración. Incluir todos los cuantificadores necesarios (los cuantificadores son los  $\forall s$  y los  $\exists s$ ).

Por inducción en  $e$ , vamos a probar:

$$\forall e :: \text{Expr} \cdot P(e)$$

Donde:

$$P(e) = \text{cantLit } e = S (\text{cantOp } e)$$

En función del Lema de Generación para Expr, tenemos que probar dos casos base y 4 casos inductivos:

- Caso base Const
- Caso base Rango
- Caso inductivo Suma
- Caso inductivo Resta
- Caso inductivo Mult
- Caso inductivo Div

## C – Demostración

Demostrar los casos correspondientes a los casos base y al constructor Suma. Los demás casos inductivos son análogos a este último, y por eso les pedimos que no los escriban para este trabajo práctico. En general en la materia siempre tendrán que escribir todos los casos, aunque sean análogos o similares, excepto que les digamos explícitamente que no es necesario.

Reproducimos las ecuaciones de las que disponemos y agregamos un conjunto de ecuaciones que corresponden con los casos del constructor de Expr, para facilitar las demostraciones:

```
data Expr = Const Float
          | Rango Float Float
          | Suma Expr Expr
          | Resta Expr Expr
          | Mult Expr Expr
          | Div Expr Expr
  deriving (Show, Eq)

data Nat = Z | S Nat
```

```

suma :: Nat → Nat → Nat
suma Z m          = m          -- {S1}
suma (S n) m      = S (suma n m) -- {S2}

cantLit :: Expr → Nat
cantLit (Const _) = S Z        -- {L1}
cantLit (Rango _ _) = S Z      -- {L2}
cantLit (Suma a b) = suma (cantLit a) (cantLit b) -- {L3}
cantLit (Resta a b) = suma (cantLit a) (cantLit b) -- {L4}
cantLit (Mult a b) = suma (cantLit a) (cantLit b) -- {L5}
cantLit (Div a b) = suma (cantLit a) (cantLit b) -- {L6}

cantOp :: Expr → Nat
cantOp (Const _) = Z          -- {O1}
cantOp (Rango _ _) = Z        -- {O2}
cantOp (Suma a b) = S (suma (cantOp a) (cantOp b)) -- {O3}
cantOp (Resta a b) = S (suma (cantOp a) (cantOp b)) -- {O4}
cantOp (Mult a b) = S (suma (cantOp a) (cantOp b)) -- {O5}
cantOp (Div a b) = S (suma (cantOp a) (cantOp b)) -- {O6}

∀n, m :: Nat ·
suma n m = suma m n          -- {CONMUT}

```

## 1 – Caso base – e = Const \_

Tenemos entonces:

```

cantLit (Const _) =
{L1}      S Z =
{O1}      S (cantOp (Const _))

```

Probamos así que para el caso en que “e = (Const \_)” vale que:

```

cantLit e = S (cantOp e)

```

## 2 – Caso base – e = Rango \_ \_

Tenemos entonces:

```

cantLit (Rango _ _) =
{L2}      S Z =
{O2}      S (cantOp (Rango _ _))

```

Probamos así que para el caso en que “e = (Rango \_ \_)” vale que:

```

cantLit e = S (cantOp e)

```

### 3 – Caso inductivo – e = Suma a b

#### Hipótesis Inductiva:

{HIa}             $P(a) = \text{cantLit } a = S (\text{cantOp } a)$   
{HIb}             $P(b) = \text{cantLit } b = S (\text{cantOp } b)$

#### Tesis Inductiva:

$P(e) = \text{cantLit } e = S (\text{cantOp } e)$

#### Demostración:

$\text{cantLit } (\text{Suma } a \ b) =$   
{L3}             $\text{suma } (\text{cantLit } a) (\text{cantLit } b) =$   
{HIa, HIb}       $\text{suma } (S (\text{cantOp } a)) (S (\text{cantOp } b)) =$   
{S2}             $S (\text{suma } (\text{cantOp } a) (S (\text{cantOp } b))) =$   
{CONMUT}       $S (\text{suma } (S (\text{cantOp } b)) (\text{cantOp } a)) =$   
{S2}             $S (S (\text{suma } (\text{cantOp } b) (\text{cantOp } a))) =$   
{CONMUT}       $S (S (\text{suma } (\text{cantOp } a) (\text{cantOp } b))) =$   
{O3}             $S (\text{cantOp } (\text{Suma } a \ b))$

Probamos así que para el caso en que “e = (Suma a b)” vale que:

$\text{cantLit } e = S (\text{cantOp } e)$

### 4 – Caso inductivo Resta

No se pidió demostrar este caso, pero la lógica es similar al caso de Suma, reemplazando el uso de las ecuaciones L3 y O3 por L4 y O4 respectivamente.

### 5 – Caso inductivo Mult

No se pidió demostrar este caso, pero la lógica es similar al caso de Suma, reemplazando el uso de las ecuaciones L3 y O3 por L5 y O5 respectivamente.

### 6 – Caso inductivo Div

No se pidió demostrar este caso, pero la lógica es similar al caso de Suma, reemplazando el uso de las ecuaciones L3 y O3 por L6 y O6 respectivamente.

## Enunciado Ejercicio 12

Necesitamos demostrar que toda expresión tiene un literal más que su cantidad de operadores. Los literales son las constantes y los rangos.

Para esto se dispone de las siguientes definiciones <sup>2</sup>:

```
data Nat = Z | S Nat

suma :: Nat → Nat → Nat
suma Z m          = m                -- {S1}
suma (S n) m      = S (suma n m)    -- {S2}

cantLit :: Expr → Nat
cantLit (Const _)      = S Z          -- {L1}
cantLit (Rango _ _)    = S Z          -- {L2}
cantLit (Suma a b)      = suma (cantLit a) (cantLit b) -- {L3}
cantLit (Resta a b)     = suma (cantLit a) (cantLit b) -- {L4}
cantLit (Mult a b)      = suma (cantLit a) (cantLit b) -- {L5}
cantLit (Div a b)       = suma (cantLit a) (cantLit b) -- {L6}

cantOp :: Expr → Nat
cantOp (Const _)      = Z             -- {O1}
cantOp (Rango _ _)    = Z             -- {O2}
cantOp (Suma a b)      = S (suma (cantOp a) (cantOp b)) -- {O3}
cantOp (Resta a b)     = S (suma (cantOp a) (cantOp b)) -- {O4}
cantOp (Mult a b)      = S (suma (cantOp a) (cantOp b)) -- {O5}
cantOp (Div a b)       = S (suma (cantOp a) (cantOp b)) -- {O6}
```

La propiedad a demostrar queda expresada de la siguiente manera:

$$\forall e :: \text{Expr} \cdot \text{cantLit } e = S (\text{cantOp } e)$$

Se pide:

- A. Definir el predicado unario correspondiente a una demostración por inducción estructural (¿en qué estructura?) de esta propiedad.
- B. Definir el esquema formal de inducción estructural correspondiente a dicha demostración. Incluir todos los cuantificadores necesarios (los cuantificadores son los  $\forall$ s y los  $\exists$ s).
- C. Demostrar los casos correspondientes a los casos base y al constructor Suma. Los demás casos inductivos son análogos a este último, y por eso les pedimos que no los escriban para este trabajo práctico. En general en la materia siempre tendrán que escribir todos los casos, aunque sean análogos o similares, excepto que les digamos explícitamente que no es necesario.
  - Todos los pasos de la demostración deben estar debidamente justificados usando las herramientas que vimos en clase.
  - Pueden asumir el siguiente lema como válido. No hace falta demostrarlo:

$$\{\text{CONMUT}\} \quad \forall n, m :: \text{Nat} \cdot \text{suma } n \text{ } m = \text{suma } m \text{ } n$$

<sup>2</sup> Estas funciones están definidas usando pattern matching y recursión explícita para facilitar la demostración, sin embargo no dejen de practicar cómo sería una demostración donde estén definidas usando foldExpr.