

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

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 ²:

```

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:

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

² 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*.

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 será:

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

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:

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

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:

```

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 Const

En este caso, asumimos que

```
{CONST}      e = Const _
```

Tenemos entonces:

```

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

```

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

```
cantLit e = S (cantOp e)
```

2 – Caso base Rango

En este caso, asumimos que

```
{RANGO}      e = Rango _ _
```

Tenemos entonces:

```

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

```

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

```
cantLit e = S (cantOp e)
```

3 – Caso inductivo Suma

En este caso, asumimos que

```
{SUMA}      e = Suma a b
```

Hipótesis Inductiva:

```

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

```

Tesis Inductiva:

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

Demostración:

```

cantLit e =
{SUMA}      cantLit (Suma a b) =
{L3}        suma (cantLit a) (cantLit b) =
{HIa,HIb}   suma (S (cantOp a)) (S (cantOp b)) =
{S2}        S (suma (cantOp a) (S (cantOp b)) ) =
{CONMUT}    S (suma ( S (cantOp b) ) (cantOp a) ) =
{S2}        S (S (suma (cantOp b) (cantOp a) ) ) =
{CONMUT}    S (S (suma (cantOp a) (cantOp b) ) ) =
{O3}        S (cantOp (Suma a b) ) =
{SUMA}      S (cantOp e)

```

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

```
cantLit e = S (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 SUMA, L3 o O3 por RESTA, L4 y O4, donde RESTA se define como:

`{RESTA} e = Resta a b`

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 SUMA, L3 o O3 por MULT, L5 y O5, donde MULT se define como:

`{MULT} e = Mult a b`

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 SUMA, L3 o O3 por DIV, L6 y O6, donde DIV se define como:

`{DIV} e = Div a b`