Esquema de Inducción

Grupo cNil

2025-09-16

Ejercicio 12

Descripción del problema de demostracion

Necesitamos demostrar que *toda la 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.

```haskell
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}
```

Y del siguiente lema que podemos asumir como válido.
No hace falta demostrarlo:

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

Dado que `cantList` y `cantOp` reciben un tipo de dato `Expr` haríamos bien en recordar cómo está compuesta su estructura.

Expr

```
data Expr = Const Float
          | Rango Float Float
          | Suma Expr Expr
          | Resta Expr Expr
          | Mult Expr Expr
          | Div Expr Expr
```

Propiedad a Demostrar $\forall e :: Expr \ \cdot \ cantLit \ \ e = S \ \ (cantOp \ \ e)$

Demostración

a) Predicado Unario Dado que la propiedad opera sobre expresiones `Expr` tiene sentido definir el *predicadio unario* correspondiente a la demostracion por induccion estructural en una expresión `e :: Expr`. Queda definido como: $P(e) := \ cantList \ \ e = S \ \ (cantOp \ \ e)$

b) Esquema formal de induccion estructural Por el principio de inducción estructural sobre `Expr` [**declarar teorema**], **si**

$$
\begin{aligned}
&(\forall x :: Float.\ P(Const\ x) \\
\wedge\ \ & \forall\ x :: Float.\ \forall\ y :: Float.\ P(Rango\ x\ y) \\
\wedge\ \ & \forall\ e1 :: Expr.\ \forall\ e2 :: Expr.\ ((P(e1)\ \wedge\ P(e2))\ \rightarrow\ Suma\ e1\ e2) \\
\wedge\ \ & \forall\ e1 :: Expr.\ \forall\ e2 :: Expr.\ ((P(e1)\ \wedge\ P(e2))\ \rightarrow\ Resta\ e1\ e2) \\
\wedge\ \ & \forall\ e1 :: Expr.\ \forall\ e2 :: Expr.\ ((P(e1)\ \wedge\ P(e2))\ \rightarrow\ Mult\ e1\ e2) \\
\wedge\ \ & \forall\ e1 :: Expr.\ \forall\ e2 :: Expr.\ ((P(e1)\ \wedge\ P(e2))\ \rightarrow\ Div\ e1\ e2))
\end{aligned}
$$

entonces $\forall e :: Expr.\ P(e)$.

c) Demostración

```
Const Float -- Caso Base
```

$$
\begin{aligned}
&\forall\ x :: Float.\ P(Const\ x) := \\
&cantList\ (Const\ x) = S\ (cantOp\ (Const\ x))
\end{aligned}
$$

```
cantList (Const x) = S Z -- {L1} por un lado
S cantOp (Const x) = S Z -- {O1} por otro.
```

□

```
Rango Float Float -- Caso Base
```

$$\forall\ x :: Float.\ \forall\ y :: Float.\ P(Rango\ x\ y) :=$$
$$cantList\ (Rango\ x\ y) = S\ (cantOp\ (Rango\ x\ y))$$

```
cantList (Rango x y) = S Z -- {L2} por un lado
S cantOp (Rango x y) = S Z -- {O2} por otro.
```

□

```
Suma Expr Expr -- Caso Inductivo
```

$\forall\ e1 :: Expr.\ \forall\ e2 :: Expr.\ ((P(e1)\ \land\ P(e2))\ \rightarrow\ Suma\ e1\ e2)$
$donde\ P(Suma\ e1\ e2) := cantList\ (Suma\ e1\ e2) = S\ (cantOp\ (Suma\ e1\ e2))$

```
cantList (Suma e1 e2) = suma (cantList e1) (cantList e2)        -- {L3}
                      = suma (S (cantOp e1)) (S (cantOp e2))    --  HI
                      = S (suma (cantOp e1)) (S (cantOp e2))    -- {S2}
                      = S (suma (S (cantOp e2)) (cantOp e1))    -- {CONMUT}
                      = S (S (suma (cantOp e2) (cantOp e1)))    -- {S2}
                      = S (S (suma (cantOp e1) (cantOp e2)))    -- {CONMUT}
                      = S (cantOp (Suma e1 e2))                 -- {O3}
                      -- Como se quería probar.
```

□

Los demás casos inductivos son análogos a este último (cambiar “Suma” por “Mult”, “Resta” y “Div” usando {L4}, {L5}, {L6} y {O4}, {O5}, {O6} en vez de {L3} y {O3} respectivamente en cada caso).

Por lo tanto. $\forall\ e :: Expr.\ P(e)$.

■