12

```
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                                     -- {01}
cantOp (Rango _ _) = Z                                     -- {02}
cantOp (Suma a b)  = S (suma (cantOp a) (cantOp b))        -- {03}
cantOp (Resta a b) = S (suma (cantOp a) (cantOp b))        -- {04}
cantOp (Mult a b)  = S (suma (cantOp a) (cantOp b))        -- {05}
cantOp (Div a b)   = S (suma (cantOp a) (cantOp b))        -- {06}
```

La propiedad a demostrar queda expresada de la siguiente manera:

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

data Expr = Const Float | Rango Float Float | Suma Expr Expr | Resta Expr Expr | Div Expr Expr | '

Quiérenos demostrar que

$$\forall e :: Expr.\ cantLit\ e = S\ (cantOp\ e)$$

Como la propiedad le esta aplicando funciones a una única variable e que es del tipo Expr, vamos a hacer inducción sobre la estructura Expr.

a) $P(e) = cantLit\ e = S\ (cantOp\ e)$

Para demostrar que la propiedad vale para todo e de la forma Expr, tenemos que demostrar que vale para los constructores base al igual que los recursivos, en el caso de los recursivos, asumiendo como hipótesis inductiva que la proposición vale para sus parámetros. Como la consigna nos pide que solo demostremos el constructor recursivo suma, asumimos que la proposición vale para el resto de los constructores recursivos

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

CB I: $\forall x :: Float . P(Const\ x)$

CB II: $\forall x\ y :: Float . P(Rango\ x\ y)$

Paso Inductivo:

.) $\forall a, b :: Expr . P(a) \wedge P(b) \Rightarrow P(Suma\ a\ b)$
.) $\forall a, b :: Expr . P(a) \wedge P(b) \Rightarrow P(Resta\ a\ b)$
.) $\forall a, b :: Expr . P(a) \wedge P(b) \Rightarrow P(Mult\ a\ b)$
.) $\forall a, b :: Expr . P(a) \wedge P(b) \Rightarrow P(Div\ a\ b)$

CB I | $\forall q :: Float$

$$CantLit(Const\ q) = S(CantOp\ Const\ q)$$

{L1B

$$S(Z) = S(Z)\ \surd$$

{01

CB II | $\forall a, b :: Float$

$$CantLit(Rango\ a\ b) = S(cantOp(Rango\ a\ b))$$

{L2} $S\ Z = S(Z)$ ✓ {O2}

Paso Inductivo: $e = Suma\ x\ y \forall x, y :: Expr$

$$CantLit(Suma\ x\ y) = S(cantOp(Suma\ x\ y))$$

{O3}

{L_3} → $suma\ (cantLit\ x)(cantLit\ y) = S(S(suma\ (cantOp\ x)(cantOp\ y)))$

{HI} → $suma\ (S(cantOp\ x))(\underbrace{S(cantOp\ y)}_{m}) = S(S(suma\ (cantOp\ x)\ (cantOp\ y)))$

{S_2} → $S(suma\ (cantOp\ x)(S(cantOp\ y))) = S(S(suma\ (cantOp\ x)(cantOp\ y)))$

{Commut}

$S(suma\ S(cantOp\ y)(cantOp\ x)) = S(S(suma\ (cantOp\ x)(cantOp\ y)))$

{S_2} → $S(S(suma\ (cantOp\ y)(cantOp\ x))) = S(S(suma\ (cantOp\ x)\ (cantOp\ y)))$

{Commut}

$S(S(suma\ (cantOp\ x)(cantOp\ y))) = S(S(suma\ (cantOp\ x)\ (cantOp\ y)))$.

⇒ Luego, al cumplir la igualdad del p.i., queda demostrada $\forall e :: Expr.\ P(Expr\ e)$