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:

1. Enunciado

```
data Nat = Z | S Nat


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

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}
```

Nos piden demostrar la siguiente afirmación

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

2. Lema

Vamos primero a plantear el siguiente lema:

$$\forall \texttt{n :: Nat}, \forall \texttt{m :: Nat. suma (S n) (S m) = S(S(suma n m))}$$

Lo demostramos por Inducción sobre Naturales:

Planteamos el predicado unario

$$\texttt{P(n) = }\forall\texttt{m::Nat. suma (S n) (S m) = S(S(suma n m))}$$

Queremos demostrar que $\forall$`n::Nat.P(n)` vale.

Por el principio de inducción sobre n, tenemos que probar que:

Vale `P(Z)` (caso base).
$\forall$`n :: Nat. P(n')` $\implies$ `P(S n')`(*paso inductivo*).

Vamos a probar que `P(Z)` $=$ $\forall$`m :: Nat. suma (S Z) (S m) = S(S(suma Z m))` **(Caso Base)** :

$$\begin{aligned}
\texttt{suma (S Z) (S m)} &= \texttt{S(S(suma Z m))} & \\
\texttt{S(suma Z (S m))} &= \texttt{S(S(suma Z m))} & \{S2\} \\
\texttt{S(S m)} &= \texttt{S(S(suma Z m))} & \{S1\} \\
\texttt{S(S m)} &= \texttt{S(S m)} & \{S1\}
\end{aligned}$$

✓**cumple el caso base**

Paso inductivo `n = S n'`

Queremos probar que $\forall$`n':: Nat. P(n')` $\implies$ `P(S n')`.

Asumimos que `P(n')` vale, o sea que

$$\texttt{P(n') = }\forall\texttt{m::Nat.suma (S n') (S m) = S(S(suma n' m))} \qquad \{HI\}$$

Vamos a probar que `P(S n')` $=$ $\forall$`m :: Nat. suma (S(S n') (S m) = S(S(suma (S n') m))`

$$\begin{aligned}
\texttt{suma (S(S n')) (S m)} &= \texttt{S(S(suma (S n') m))} & \\
\texttt{S(suma (S n') (S m))} &= \texttt{S(S(suma (S n') m))} & \{\text{S2}\} \\
\texttt{S(S(S(suma n' m)))} &= \texttt{S(S(suma (S n') m)))} & \{\text{HI}\} \\
\texttt{S(S(S(suma n' m)))} &= \texttt{S(S(S(suma n' m)))} & \{\text{S2}\}
\end{aligned}$$

✓**cumple el paso inductivo**

Dado que probamos el caso base y el paso inductivo, por Induccion en Naturales, nuestro Lema vale para todo `n :: Nat`.

3. Demostración de la propiedad

Recordemos la propiedad a demostrar

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

Vamos a demostrarla usando inducción sobre e.

Planteamos el predicado unario: `P(e)= cantLit  e = S (cantOp e)`

Queremos probar que $\forall$`e:: Expr. P(e)`

`Expr.` tiene varios constructores pero solo vamos a demostrar la propiedad para `Const a`, `Rango a b.` y `Suma e f` con `a, b::Float` y `e, f ::Expr`. Los demas constructores son analogos a `Suma`.

Por el principio de inducción sobre e, tenemos que probar que:

Vale el primer caso base `P(Const a). a:: Float`
Vale el segundo caso base `P(Rango a b). a,b:: Float`
$\forall$`e, f:: Expr. P(e) ^ P(f)` $\Longrightarrow$ `P(Suma e f).`

Probemos los casos base `e= Const a. a::Float`:

$$\begin{aligned}\texttt{cantLit (Const a)} &= \texttt{S (cantOp (Const a)} & \\ \texttt{S Z} &= \texttt{S (cantOp (Const a)} & \{\text{L1}\} \\ \texttt{S Z} &= \texttt{S Z} & \{\text{O1}\}\end{aligned}$$

✓**cumple este caso base**

Caso `e= Rango a b. a,b::Float`

$$\begin{aligned}\texttt{cantLit (Rango a b)} &= \texttt{S (cantOp (Rango a b)} & \\ \texttt{S Z} &= \texttt{S (cantOp (Rango a b)} & \{\text{L2}\} \\ \texttt{S Z} &= \texttt{S Z} & \{\text{O2}\}\end{aligned}$$

✓**cumple este caso base**

Paso inductivo, `e = Suma f g. f, g:: Expr`

Vamos a ver que $\forall$`e', f':: Expr.  P(e') ^ P(f')` $\Longrightarrow$ `P(Suma e' f')`

Suponemos que `P(e')` y `P(f')` valen. Es decir :

`cantLit e' = S (cantOp e')`   {HI 1}
`cantLit f' = S (cantOp f')`   {HI 2}

Queremos ver que `P(e)`

$$
\begin{aligned}
\texttt{cantLit (Suma e' f')} &= \texttt{S (cantOp (Suma e' f'))} \\
\texttt{suma (cantLit e') (cantLit f')} &= \texttt{S (cantOp (Suma e' f'))} \quad \{\text{L3}\} \\
\texttt{suma (S (cantOp e')) (cantLit f')} &= \texttt{S (cantOp (Suma e' f'))} \quad \{\text{H1}\} \\
\texttt{suma (S (cantOp e')) (S (cantOp f'))} &= \texttt{S (cantOp (Suma e' f'))} \quad \{\text{H2}\} \\
\texttt{S (S (suma (cantOp e') (cantOp f')))} &= \texttt{S (cantOp (Suma e' f'))} \quad \{\text{Lema}\} \\
\texttt{S (S (suma (cantOp e') (cantOp f')))} &= \texttt{S (S (suma (cantOp e') (cantOp f')))} \quad \{\text{O3}\} \\
&\checkmark\textbf{cumple el paso inductivo}
\end{aligned}
$$

Como probamos todos los casos base y los pasos inductivos correspondientes, por el principio de Induccion sobre el tipo `Expr` vale que ∀ `e :: Expr. P(e)`.