Assignment 4

Paul Vrbik : 301056796 : CMPT 731 : Fall 2006

November 10, 2006

Series Multiplication :

For two series, `Series as` and `Series bs`, the nth term of `Series as * Series bs` is given by

$$\Theta_n = a_n b_0 + a_{n-1} b_1 + a_{n-2} b_2 + ... + a_1 b_{n-1} + a_0 b_n$$

where $a_n =$ `as!!n` and $b_n =$ `bs!!n`.

Implemented as:

```
Series as * Series bs = Series (map (r as bs) [0..]) where
    r (z:zs) (y:ys) 0 = z*y
    r (zs) (y:ys) m = (zs!!m)*y + (r zs ys (m-1))
```

Series Division :

For two series `Series as` and `Series bs` the nth term of `Series as / Series bs` is given by

$$\Pi_n = \frac{1}{b_o} \cdot \left(a_n - \sum_{i=1}^{n} (b_i \cdot \Pi_{n-i}) \right)$$

where $a_n =$ `as!!n` and $b_n =$ `bs!!n`.

This has (nontrivially effficient) implementation as follows:

```
Series (pi2 (as) []) where
    pi2 (d:ds) l = (piCalc) : (pi2 ds (piCalc:l)) where
        piCalc = (1/b) * (d - (foldr (+) 0 (zipWith (*) l bs) ))
```

This definition is not robust enough to deal with divisions by zeros caused by evaluating $\frac{1}{x}$ for instance. We output an exception in this case. For equations like $\frac{x+x^2+x^3}{x+2x^2+x^3}$ where an x can be factored and cancelled we check to see if both constants are 0. If this is the case we simply evaluate $\frac{1+x+x^2}{1+2x+x^2}$ instead. These changes are reflected in the code:

```
Series (a:as) / Series (b:bs)
    | and[(a==0),(b==0)]    = Series as / Series bs
    | (b==0)                = error "Division by zero"
    | otherwise             = Series (pi2 (a:as) []) where
           pi2 (d:ds) l = (piCalc) : (pi2 ds (piCalc:l)) where
               piCalc = (1/b) * (d - (foldr (+) 0 (zipWith (*) l bs) ))
```

The Code :

```
module Series (Series(),
               Ratio(..),
               Complex(..),
               x,
               integrate,
               sine,
               cosine,
               expo,
               evaluate
              ) where

import Monad
import Ratio
import Complex

newtype (Fractional a) => Series a = Series [a] deriving (Eq)

fibs = unSeries (1/(1-x-x^2))

unSeries :: (Fractional a) => Series a -> [a]
unSeries (Series as) = as

instance (Fractional a) => Num (Series a) where
    Series as + Series bs = Series (zipWith (+) as bs)
    negate (Series as) = Series (map (negate) as)
    fromInteger n = Series (fromRational (n % 1):repeat 0)
    Series as * Series bs = Series (map (r as bs) [0..]) where
        r (z:zs) (y:ys) 0 = z*y
        r (zs) (y:ys) m = (zs!!m)*y + (r zs ys (m-1))

instance (Fractional a) => Fractional (Series a) where
    fromRational a = Series (fromRational a:repeat 0)
    Series (a:as) / Series (b:bs)
        | and[(a==0),(b==0)]    = Series as / Series bs
        | (b==0)                = error "Division by zero"
        | otherwise             = Series (pi2 (a:as) []) where
                pi2 (d:ds) l = (piCalc) : (pi2 ds (piCalc:l)) where
                    piCalc = (1/b) * (d - (foldr (+) 0 (zipWith (*) l bs) ))

x :: (Fractional a) => Series a
x = Series $ 0 : 1 : repeat 0
```

```
instance (Fractional a) => Show (Series a) where
    show (Series (a:bs))
       = show a ++ concat (zipWith term bs [1..maxPower]) ++ " + ..."
         where
         maxPower = 20
         term coeff power =
            let c_string@(c:cs) = show coeff
                p_string = if power == 1 then "*x" else "*x^" ++ show power
            in if coeff == 0 then ""
               else if c == '-' then " - " ++ cs ++ p_string
                     else " + " ++ c_string ++ p_string

integrate :: (Fractional a) => a -> Series a -> Series a
integrate constant (Series series) =
   Series $ constant : zipWith (/) series (iterate (1+) 1)

evaluate :: (Fractional a, Show a) => Series a -> a -> a
evaluate (Series as) x =
   best . scanl (+) 0 . zipWith (*) as . iterate (*x) $ 1
   where
   best = last . take 100 . limit
   limit (x:xs@(y:ys@(z:zs))) = if x==y && y==z then [x] else x:limit xs

sine, cosine, expo :: (Fractional a) => Series a
sine = integrate 0 cosine
cosine = integrate 1 (0 - sine)
expo = integrate 1 expo
```

Test Cases :

```
*Series> 1/(1-x-x^2)
1.0 + 1.0*x + 2.0*x^2 + 3.0*x^3 + 5.0*x^4 + 8.0*x^5 + 13.0*x^6 + 21.0*x^7 +
34.0*x^8 + 55.0*x^9 + 89.0*x^10 + 144.0*x^11 + 233.0*x^12 + 377.0*x^13 +
610.0*x^14 + 987.0*x^15 + 1597.0*x^16 + 2584.0*x^17 + 4181.0*x^18 +
6765.0*x^19 + 10946.0*x^20 + ...

*Series> map (evaluate (sine/cosine)) (map (*(pi/12))[0..12])
[0.0,0.2679491924311227,0.5773502691896258,0.9999999999999998,
1.7320508075688763,3.7320507471911735,62.70704757820666,
1.4962402458085382e7,3.8281549864470127e12,2.7607987658236122e17,
6.577469719842826e21,6.204342101292763e25,2.690038121863066 4e29]

*Series> map (evaluate sine) (map (*(pi/12)) [0..12])
[0.0,0.2588190451025207,0.49999999999999994,0.7071067811865475,
0.8660254037844386,0.9659258262890681,1.0000000000000002,
0.9659258262890684,0.8660254037844392,0.7071067811865477,
0.5000000000000008,0.2588190451025209,-9.735493367934764e-17]

*Series> sine/cosine
0.0 + 1.0*x + 0.33333333333333337*x^3 + 0.13333333333333336*x^5 +
5.396825396825398 5e-2*x^7 + 2.186948853615521e-2*x^9 +
8.863235529902199e-3*x^11 + 3.592128036572482e-3*x^13 +
1.4558343870513185e-3*x^15 + 5.900274094558 61e-4*x^17 +
2.391291142435525e-4*x^19 + ...

*Series> map (evaluate expo) (map (*((0:+1)*pi/12)) (take 13 $ iterate (1+) 0))
[1.0 :+ 0.0,0.9659258262890684 :+ 0.2588190451025207,0.8660254037844386 :+
0.49999999999999994,0.7071067811865475 :+ 0.7071067811865475,
0.5000000000000001 :+ 0.8660254037844386,0.2588190451025209 :+
0.9659258262890681,(-9.498900114615426e-18) :+ 1.0000000000000002,
(-0.25881904510252063) :+ 0.9659258262890684,(-0.4999999999999999) :+
0.8660254037844392,(-0.7071067811865477) :+ 0.7071067811865477,
(-0.8660254037844386) :+ 0.5000000000000008,(-0.9659258262890688) :+
0.2588190451025209,(-1.0000000000000009) :+ (-1.0964738041436517e-16)]

*Series> expo :: (Series (Ratio Int))
1%1 + 1%1*x + 1%2*x^2 + 1%6*x^3 + 1%24*x^4 + 1%120*x^5 + 1%720*x^6 +
1%5040*x^7 + 1%40320*x^8 + 1%362880*x^9 + 1%3628800*x^10 +
1%39916800*x^11 + 1%479001600*x^12 + 1%1932053504*x^13 +
1%1278945280*x^14 + 1%2004310016*x^15 + 1%2004189184*x^16 +
(-1)%288522240*x^17 + (-1)%898433024*x^18 + 1%109641728*x^19 +
(-1)%2102132736*x^20 + ...
```

```
*Series> 1/x
*** Exception: Division by zero

*Series> x^2/x
0.0 + 1.0*x + ...
```