PFL - Trabalho Prático 1

Representação interna

Os polinómios são uma soma de vários fatores em que cada fator é constituido por uma parte literal e um coeficiente. A parte literal pode ter zero ou vários literais em que cada literal tem um expoente associado. Deste modo, internamente, representamos a parte literal do fator como sendo uma lista de tuplos em que cada tuplo representa um literal e o seu expoente. No caso do termo independente em que a parte literal é nula, esta é representada da seguinte forma: (" ",0). De seguida, adicionamos o coeficiente e juntamos à parte literal através de um tuplo:(coef,[literal]).Por fim, juntamos todos os fatores através de uma lista: [Fator].

Normalização

O processo de normalização de um polinómio define-se como sendo uma organização por grau e,consequentemente, por literal. Por fim, todos os fatores com o mesmo literal são adicionados. Deste modo, criamos uma função sortPoli que, através de funções auxiliares, junta todos os fatores com o mesmo literal e adiciona os seus coeficientes, ordena por grau e, em caso de igualdade, por literal.

```
normalize :: Poli -> Poli
normalize polinom = [eliminate0Degree y | y<- l3]
                    where l1 = sortPoli ([(fst z, [(' ', 0)]) | z <- polinom, (snd z) == []] ++ [z | z
                          l2 = addFactor l1
                          l3 = [ x | x <- l2, (fst x) /= 0]
```

Exemplo

```
normalizeString " 2xy + 3z^2 + 6xy"
```

Soma

O processo de adição de dois polinómios define-se como sendo a junção dos mesmos e, de seguida, uma normalização deste novo polinómio. Deste modo, sendo um polinómio uma lista de fatores, concatenamos as duas listas e aplicamos a função de normalização.

```
addTwoPolis :: Poli -> Poli -> Poli
addTwoPolis pol1 pol2 = normalize (pol1 ++ pol2)
```

Exemplo

```
addTwoPolisString " 2xy + 3z^2 + 6xy" "3xyz + 7z^2"
```

Multiplicação

O processo de multiplicação de dois polinómios consiste em multiplicar todos os fatores de um polinómio com todos os fatores de outro polinómio. Deste modo, primeiramente, normalizamos ambos os polinómios, de seguida, através de dois geradores, em que cada um retira um fator de um polinómio, multiplicamos os coeficientes e concatenamos a lista de literais.Por fim, normalizamos o polinómio final.

```
multiplicatePolis :: Poli -> Poli -> Poli
multiplicatePolis pol1 pol2 = normalize [((fst x) * (fst y), (snd x) ++ (snd y)) | x <- pol3, y <- pol
                  where pol3 = normalize pol1
                        pol4 = normalize pol2
```

Exemplo

```
multiplicatePolisString " 2xy + 3z^2 + 6xy" "3xyz + 7z^2"
```

Derivação

O processo de derivação de um polinómio em função de um literal consiste em verificar todos os fatores que apresentam esse literal reduzir o seu grau por um e multiplicar o coeficiente do mesmo pelo novo grau deste literal. Deste modo, através de funções auxiliares, multiplicamos os coeficientes,reduzimos o grau do literal com var, eliminamos os fatores que não tem o literal var na lista de literais e, por fim, eliminamos todos os literais que tem grau 0.

```
derivatePoli :: Poli -> Char -> Poli
derivatePoli pol1 var = normalize [eliminate0Degree x | x <- pol5]
                        where pol5 = [x | x <- pol4, eliminateTerms x var]
                              pol4 = [reduceDegree x var | x <- pol3]
                              pol3 = multiplicateCoef pol2 var
                              pol2 = normalize pol1
```

Exemplo

```
derivatePoliString " 2xy + 3z^2 + 6xy" 'x'
```

Conversão de string para representação interna do polinómio

Para converter uma string num polinómio, primeiramente, separamos a string recebida pelos sinais "+" e "-".De seguida, removemos todos os espaços,criamos os fatores e convertemos estes para a nossa representação. Por fim, verificamos os termos independentes e adapatamos à nossa representação explicada acima.

```
createPoly :: String -> Poli
createPoly x = [(fst z, [(' ', 0)]) | z <- poly, (snd z) == []] ++ [z | z <- poly, (snd z) /= []]
            where poly = [takeCoef mono | mono <- monomyal, mono /= ""]
                  monomyal = createMonomyal strings
                  strings = removeSpaces s
                  s = splitString x
```

Conversão de representação interna do polinómio para string

Para converter um polinómio numa string percorremos a lista de fatores, adicionamos o "^" aos literais em que o grau é superior a 1 e, por fim, juntamos os "+".

```
createString :: Poli -> String
createString (x:xs) |(fst x) > 0 = "+" ++ show (fst x) ++ showLiteral (snd x) ++ createString xs
                    |otherwise = show (fst x) ++ showLiteral (snd x) ++ createString xs
createString [] = ""
```

Autores

Hugo Gomes - 202004343

João Moreira - 202005035

Grupo - G10_10