PFL_TP1_G12_02

First Project for the Course Programação Funcional e Lógica (PFL) at Faculdade de Engenharia do Porto (FEUP)

- Uma breve descrição da estratégia de implementação de cada funcionalidade;
- Exemplos de utilização que permitam testar todas as funcionalidades do programa

Representação dos Polinomios

Estrutura

Neste primeiro trabalho prático de PFL decidimos usar uma árvore binária para representar os polinómios. Nesta árvore temos 6 nós que representam os valores que podem existir num polinómio:

- ***Vazia*** : nó vazio, representa a string vazia;
- ***NoSoma*** : nó soma, representa a soma entre dois termos do polinómio;
- ***NoProd*** : nó produto, representa o produto entre dois elementos (quer sejam coeficientes ou variáveis) dentro de um termo;
- ***NoPoten*** : nó potência, representa uma variável e a sua respetiva potência;
- ***NoVar*** : nó variável, representa uma variável;
- ***NoNum*** : nó número, representa qualquer valor numérico no polinómio, seja coeficiente ou a potência de uma variável.

Dividimos sempre a estrutura em dois, na tentativa de a tornar o mais equilibrada possível no aspeto de divisão de termos.

A seguinte imagem mostra um exemplo da utilização da árvore binária, para o polinómio "3*x^2 + 7 *x + 1":

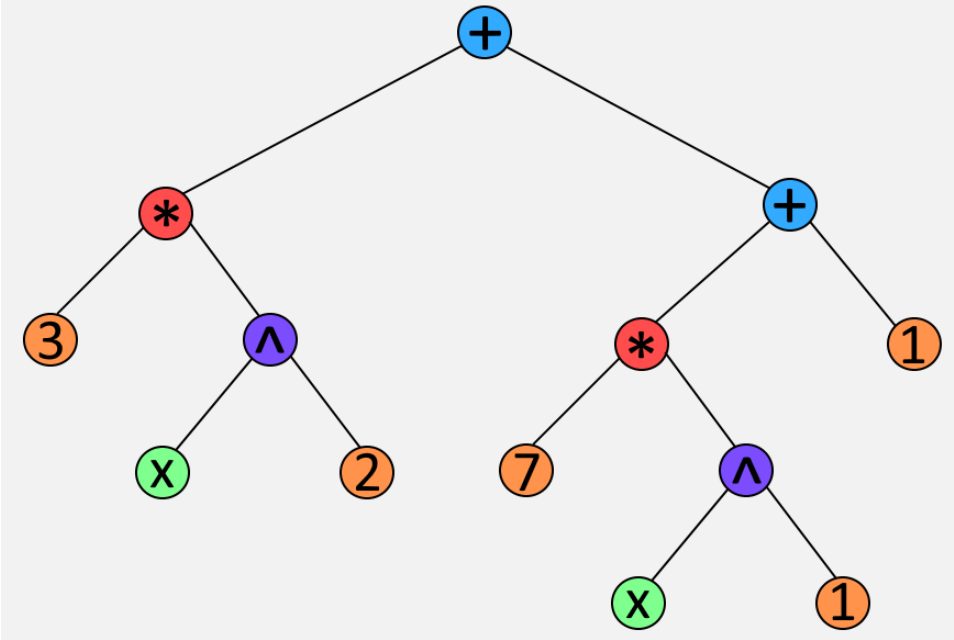

Figura 1 - Representação visual da Árvore Binária

Legenda:

- +: *NoSoma*
- *: NoProd
- ^: *NoPoten*
- Qualquer letra: *NoVar*
- Qualquer dígito: *NoNum*

Justificação

A representação do polinómio com recurso a uma árvore binária pareceu-nos substancialmente mais simples no sentido em que as separações de termos e dentro dos termos ficariam mais claras e fáceis de atravessar.

Tal como foi mostrado na figura 1, os termos são separados por *NoSoma*'s, pelo que se torna mais simples obter os termos.

Procurando recursivamente por *NoSoma*'s, quando já não for encontrado um *NoSoma* significa que chegamos a um dos termos do polinómio.

Os coeficientes e variáveis, quando dentro de um termo são claramente diferenciáveis pelos *NoProd*'s. Procurando por *NoProd*'s, quando já não for encontrado um *NoProd* implica 1 de duas possibilidades:

- Encontra *NoNum*: que é um valor numérico e representa um coeficiente;
- Encontra *NoPoten*: que é uma variável e a sua respetiva potência.

A maior **vantagem** seria a operação de derivar, já que se tornou tão simples ao ponto de ser feita em apenas 1 função. Bastava trocar um *NoPoten* por um *NoProd* que multiplicava o expoente pela variavel elevada ao (expoente-1).

A maior **desvantagem** seria a normalização do polinómio, já que obrigou à utilização de várias funções auxiliares e algumas um pouco rebuscadas.

Funcionalidades

Normalizar um Polinómio - normPoly :: String -> String

> Na função normPoly recebemos uma string que representa um polinómio.
> Primeiro transformamos a string numa árvore binária do tipo já apresentado.
> De seguida encontramos todas as variáveis existentes no polinómio e colocamo-las numa lista ordenada.
> Depois acumulamos numa lista (com o tamanho da anterior mas inicialmente preenchida com 0's) os coeficientes no índice correspondente às variáveis da lista anterior.
> Finalmente concatenamos o polinómio na forma de uma string.
> Retornamos uma string com o polinómio normalizado.

Soma de Polinómios - sumPoly :: String -> String -> String

> Na função sumPoly recebemos 2 strings, cada uma representando um polinómio.
> Simplesmente concatenamos as strings com o carácter '+' e deixamos a normalização fazer o resto.
> Retornamos uma string com o resultado da soma normalizado.

Produto de Polinómios - multPoly :: String -> String -> String

> Na função multPoly recebemos 2 strings, cada uma representando um polinómio.
> Primeiro separamos os termos dos dois polinómios em 2 listas, uma lista para cada conjunto de termos.
> De seguida, juntamos cada termo da primeira lista com todos os termos da segunda lista através de um *NoProd*.
> Finalmente, juntamos a lista resultante com *NoSoma* e deixamos a normalização fazer o resto.
> Retornamos uma string com o resultado do produto normalizado.

Derivar um Polinómio - derivPoly :: String -> String -> String

> Recebemos uma string que representa o polinómio e uma segunda string que representa a variável a ser derivada.
> Simplesmente atravessamos recursivamente todos os nós:
>
> - Se encontrar um *NoProd*:
>   - se encontra um *NoPoten* com um *NoVar* que seja igual à variável a ser derivada, troca o *NoPoten* por um *NoProd* que multiplica o expoente por um *NoPoten* com a variável elevada ao (expoente-1);
>   - senão ignora o *NoNum*, que é a única outra opção que pode aparecer;
> - se encontrar um *NoNum* retorna zero;
> - senão continua a travessia recursiva através dos *NoSoma*'s.
>
> Finalmente normalizamos a árvore resultante.
> Retornamos uma string com o resultado da derivada normalizada.