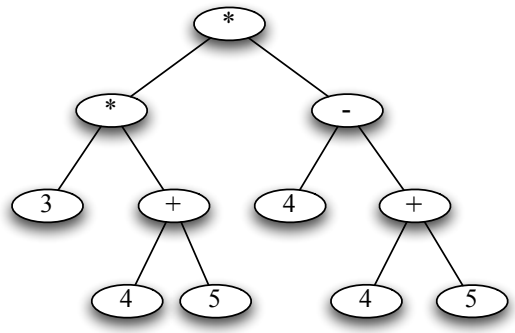
Aula Teórico-prática 13

Programação Funcional

LEI 1º ano

Relembre o exercício da ficha 9 onde, a partir de uma expressão (`ExpInt`) se produzia a correspondente string em notação *posfixa*. A expressão $(3*(4+5))*(4-(4+5))$, representada pela árvore



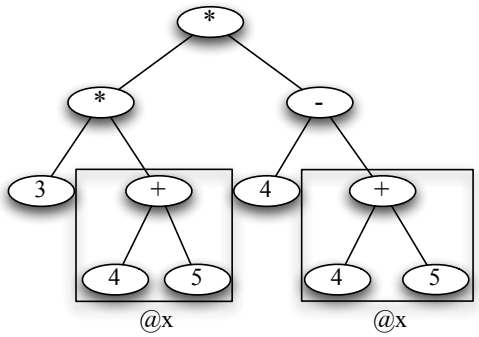
era convertida na string `"3 4 5 + * 4 4 5 + - *"`.

1. O objectivo deste exercício é agora definir o processo inverso: a obtenção da expressão (do tipo `ExpInt`) a partir da sua representação posfixa. Use para isso a função `lex :: String -> [(String,String)]` que dá como resultado uma lista de pares `(prefixo,sufixo)` da string argumento. Use ainda o facto de uma das instâncias pré-definidas da classe `Read` serem os números inteiros (e por isso a função `read` estar definida para `String -> Int`).

2. Use a função anterior para definir uma função `calculaP :: String -> Int` que calcule o valor de uma expressão em posfixo.

3. Considere agora uma extensão à sintaxe posfixa de forma a permitir *"memorizar"* alguns resultados intermédios. Usaremos para isso o sinal `@` seguido do nome de uma variável.

   Por exemplo, a expressão `"3 4 5 + @ x * 4 x - *"` corresponde à árvore



   (a) Redefina a função de leitura de expressões de forma a contemplar esta extensão.

   (b) Redefina ainda a função de cálculo destas expressões, tendo em consideração que cada expressão memorizada só deve ser calculada uma única vez.