**No g1** foi só devolver a probabilidade.

**No g2:**

A expressão prod (a Empty) (b Empty) é uma distribuição de pares de LTree a, ou seja, do tipo Dist (Ltree A, LTree A). Mas nós queremos Dist (LTree A) mas, a única função que transforma duas LTree’s A em LTree A é o Fork.

Mas... existe um problema. Sendo que esse problema é que, a função Fork não sabe o que fazer com distribuições, mas sabe o que fazer com pares de LTrees.

Mas para isto funcionar precisamos primeiro perceber como fazer com que Fork retorne Dist (LTree a) em vez de LTree a

De alguma maneira, tivemos que criar uma nova função baseada em Fork que nos permita receber distribuições. Para tal, tivemos que usar maquinaria monádica.

Então, ao assumirmos que Fork é do tipo (LTree a, LTree a) --> Dist (LTree a) conseguiríamos portanto definir uma função Dist (LTree a, LTree a) --> Dist (LTree a) que tem o tipo pretendido.

Mas para isto funcionar precisamos primeiro perceber como fazer com que Fork retorne Dist (LTree a) em vez de LTree a

Portanto, definimos um multiplicador “final <- prod (a Empty) (b Empty)”, em que desempacota o Dist (LTree a, LTree a) e depois aplica a função unidade após aplicar a função Fork (return . Fork) para construir o Dist (LTree a).

**Na 2ªa parte:**

Tivemos que utilizar mais uma vez a maquinaria monádica para fazer a parte recursiva da função.