Taller #1: Modelos de Programación

Profesor: Javier A. Mena Z.

15 de septiembre de 2008

Lea todo el taller antes de empezar a implementar.

1. Método de Newton Genérico

El objetivo de este ejercicio es implementar el método de Newton para hallar raices de (casi) cualquier función. Una función estará representada por una fórmula. Una fórmula es tipo de dato definido recursivamente, que puede ser:

1. un flotante.
2. el átomo `x` (que representa una variable).
3. una tupla binaria <*et*>(<*f*$_1$> <*f*$_2$>). Donde <*et*> indica la operación: `s`, `r`, `m`, `d`, `p`, para la suma, resta, multiplicación, división y potenciación respectivamente) y <*f*$_1$> y <*f*$_2$> son fórmulas.
4. la tupla unaria `l(`<*f*>`)` (la etiqueta es la letra *ele*) para el logaritmo natural.

Por ejemplo la función $f(x) = 5k + \ln(3)/(8-x)^x$ podría ser representada en Oz así:

```
s(m(5.0 k) d(l(3.0) p(r(8.0 x) x)))
```

1. Implemente una función {`Derivar F`} que dada una fórmula `F`, retorne su derivada[1] con respecto a la variable `x`. Tenga en cuenta que los números son considerados como constantes.
   Ej. Para $f(x) = 5 + 3x$, su derivada es $f'(x) = 3$, en esta implementación el resultado puede estar lleno de números innecesarios.
   {`Derivar s(5.0 m(3.0 x))`} retorna `s(0.0 s(m(0.0 x) m(3.0 1.0)))`.
2. Implemente una función {`Evaluar F V`} que dada fórmula `F` y un valor `V` flotante, calcule la evaluación de la función, evaluando `x` como `V`. Nota: La fórmula `F` siempre podrá ser evaluada.
   Ej.
   {`Evaluar d(2.0 x) x 4.0`} retorna `0.5`.

[1] Tenga en cuenta las reglas del Cuadro 1.

$$
\begin{aligned}
c &\xrightarrow{'} 0 \qquad (c \text{ es constante}) && (1)\\
x &\xrightarrow{'} 1 && (2)\\
f+g &\xrightarrow{'} f'+g' && (3)\\
f-g &\xrightarrow{'} f'-g' && (4)\\
f\cdot g &\xrightarrow{'} f'\cdot g + f\cdot g' && (5)\\
\frac{f}{g} &\xrightarrow{'} \frac{f'\cdot g - f\cdot g'}{g^2} && (6)\\
\ln(f) &\xrightarrow{'} \frac{f'}{f} && (7)\\
f^g &\xrightarrow{'} f^g\cdot\left(\frac{f'\cdot g}{f} + g'\cdot \ln f\right) && (8)
\end{aligned}
$$

Cuadro 1: Reglas de derivación usadas. f y g son funciones con respecto a x. También tenga en cuenta que $f(c) = 0$ si c es una constante.

3. Implemente una función {`Limpiar F`} que dada una formula `F`, retorne una fórmula equivalente pero que no contenga números innecesarios, como sumas con ceros, divisiones entre uno, multiplicaciones por cero y otros valores que puedan ser calculados directa y facilmente.
   Ej.
   {`Limpiar s(0.0 s(m(0.0 x) m(3.0 1.0)))`} retorna `m(3.0 1.0)` ó simplemente `3.0`.

4. Implemente una función iterativa {`Raiz F X0`} que dada una fórmula `F`, valor inicial `X0` retorne una raiz de la fórmula dada, usando el método de Newton. Se supondrá que ya se llegó a una raíz cuando la evaluar la fórmula, su valor sea muy cercano a `0.0`.

   Utilice la siguiente fórmula de Newton:

$$x_{n+1} = x_n - \frac{f(x)}{f'(x)}$$

Para probar su implementación, utilice el siguiente ejemplo, el cual calcula la fórmula $x = \sqrt[N]{V}$, es decir, $x^N - V = 0$. Esta fórmula sirve para hallar la raíz N-ésima de V.

```
fun {RaizNEsima V N}
   {Raiz r(p(x N) V) 1.0}
end
```

El llamado a {`RaizNEsima 25.0 2.0`} debe retornar 5.0 (o un número cercano).

2. Multiplicación de matrices

Es muy común el uso del computador como una herramienta para realizar operaciones matemáticas repetitivas, una de estas, es la multiplicación de matrices. El objetivo de este ejercicio es realizar una función en Oz que dadas dos matrices, retorne una matriz resultado de multiplicarlas. En Oz se puede representar una matriz como un registro `matriz(nf:NF nc:NC m:LLs)`, donde `NF` y `NC` es el número de filas y columnas respectivamente, y LLs es un lista de listas que contiene la matriz de flotantes. Así, por ejemplo, {`Nth LLs 3`} retorna tercera fila de la matriz y {`Nth {Nth LLs 3} 5`} retorna el elemento que se encuentre en la fila 3 y columna 5.

1. Construya una función {`Linea L Max`} que cree una lista de tipo `[L#1 L#2 L#3 ... L#Max]`. `L` y `Max` son enteros.

2. Use la función `Map` para crear la función {`Invertir Ls`}, donde Ls es una lista de tipo `[`A_1`#`B_1 `...`A_n`#`B_n`]` y debe retornar cada pareja invertida, es decir, una lista de tipo `[`B_1`#`A_1 `...`B_n`#`A_n`]`.

3. Use la función `List.zip` para crear la función {`ConstruirMult Xs Ys`} donde `Xs` y `Ys` son listas de parejas $Xa_i\#Xb_i$, y retorna una lista de la siguiente forma: `['*'(`Xa_1`#`Xb_1 Ya_1`#`Yb_1`) ...'*'(`Xa_n`#`Xb_n Ya_n`#`Yb_n`)]`

4. Usando `FoldR` o `FoldL`, construya la función {`MultiplicarFilas Es M1 M2`} que recibe como parámetros una lista con los elementos a multiplicar y las matrices en la forma explicada al principio del ejercicio. Todos los elementos multiplicados deben ser sumados. Cada elemento está dado de la forma `'*'(F1#C1 F2#C2)` que significa que se debe multiplicar de la matriz `M1` el elemento de la fila `F1`, columna `C1` y el elemento de la fila `F2` y columna `C2` de la matriz `M2`.

5. Construya la función {`MultiplicarMatriz M1 M2`} que retorna la multiplicación matricial de las matrices `M1` y `M2`, usando la representación descrita.

3. Notas y aclaraciones

La especificación de las funciones es *exacta*, por tanto NO CAMBIE LAS ESPECIFACIONES NI LOS ARGUMENTOS DE ENTRADA. Bajo ningún motivo ni circunstancia cambie la función de ejemplo. La función debe funcionar SIN MODIFICACIONES.

Grupos de máximo 2 personas. Se recomienda especialmente que los miembros de cada grupo no compartan información con otros grupos. Cualquier indicio de copia será considerado como fraude. Los integrantes del grupo deben estar en la capacidad de sustentar públicamente el trabajo presentado.

Debe enviar un archivo `taller1-mp-i-08.tar.gz` que contenga el codigo fuente, y un documento que explique como ejecutar los programas, a `javimena@gmail.com`. **Importante:** Colocar en el asunto del correo: [MP] Taller 1. No se aceptan archivos en formato RAR.