TALLER N°2

Evaluación 2 - Taller de Programación

Procesador Simbólico

usach

Taller de programación 1-2023

Fecha: 14/06/2023

Autor: Lucas Mesías

TALLER N°2

Procesador Simbólico

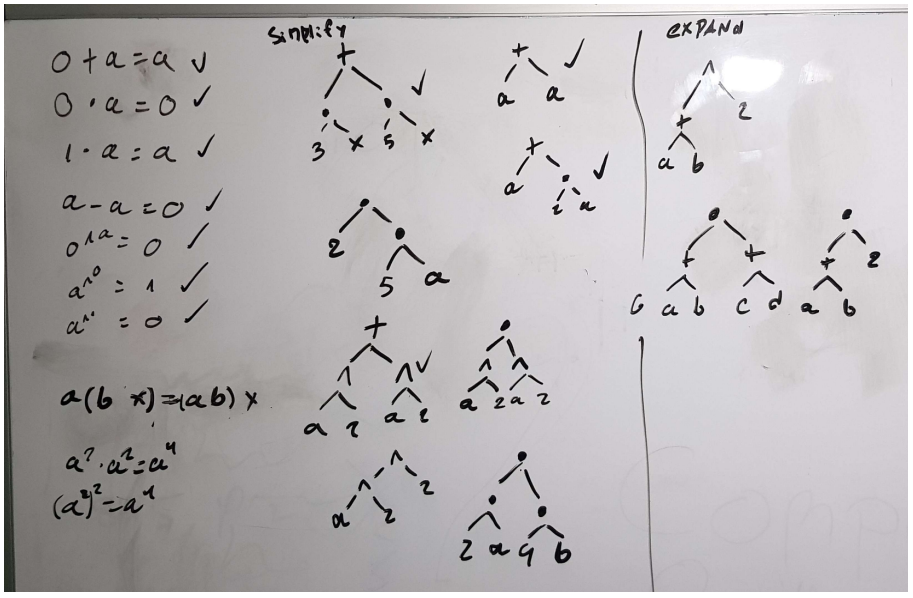
Explicación breve del algoritmo

El algoritmo crea un árbol compuesto de 3 tipos de nodos (Numero, Variable y Operación), desde una ecuación entregada en un archivo de entrada, luego dependiendo de la función aplicada a la ecuación, se opera en este. Las operaciones a realizar son Evaluar, Evaluar con reemplazo, Derivar y Simplificar.

Heurísticas o técnicas utilizadas

Cada evaluación, derivación y simplificación se realiza por casos, cada caso corresponde a reglas matemáticas.

Para cada simplificación y expansión, se dibujó el caso general matemático, y se programó el caso equivalente.



(Foto de los dibujos de expresiones que se pueden simplificar, no todas se implementaron)

Funcionamiento del programa

El programa muestra un menú donde el usuario puede elegir distintas opciones, se comienza con la lectura del archivo, del cual se crea el arbol inicial usando el código entregado por el profesor, luego se le muestra al usuario este arbol por consola, a partir de ahi, el usuario puede escoger 3 opciones: Evaluar, Evaluar con reemplazo y derivar. Simplificar no es una opción, ya que simplificar es necesario para evaluar correctamente, y para simplificar, es necesario evaluar y sería redundante separar estas operaciones. (Ejemplo: simplificar 1 + 1 y

evaluar x - x, para generar el resultado correcto en ambos casos, se debe evaluar en la función de simplificar, y simplificar en la función de evaluar)

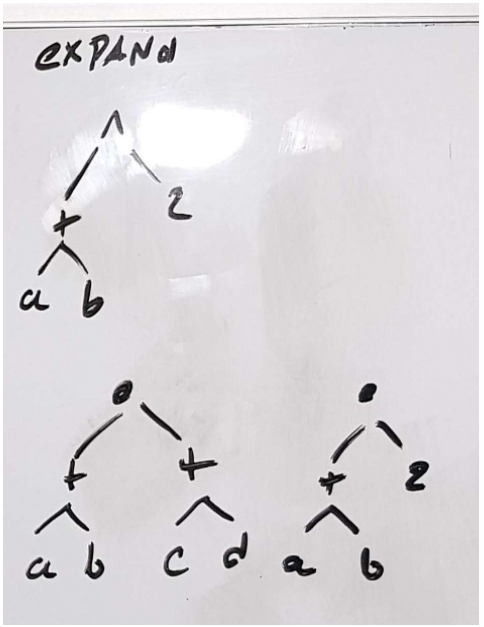
Evaluar con reemplazo: Se piden las variables a reemplazar y sus respectivos valores, luego se recorre el árbol, buscando y reemplazando estas variables, para luego usar la operación Evaluar sobre el árbol resultante.

Derivar: Se pide la variable para derivar con respecto a ella, aplicando las reglas de la derivación en cada caso correspondiente, para cada tipo de nodo, y para cada tipo de operación.

Evaluar: Esta función envoltorio expande las expresiones, simplifica el árbol de sumas y luego evalúa (función *"eval()"*) el árbol resultante.

La función *"eval()"* recorre el árbol en postorden recursivo, aplicando reglas matemáticas básicas para operaciones entre números, en caso de encontrar una variable, la operación se deja expresada, y antes de devolver el resultado recursivo de cada evaluación se simplifica la expresión (función *"simplify()"*) revisando los distintos casos implementados.

Expansión: Función no implementada, que busca operaciones "*" y "^", y revisa si son compatibles con los casos de expansión, con el propósito de transformar la mayor cantidad de expresiones a sumas de elementos no separables y trabajarlos fácilmente en la simplificación del árbol de sumas.



Dibujos de casos de expansión planeados

Simplificación del árbol de sumas: Finalmente, para resolver los árboles de sumas (por ejemplo: 1+1+1+x+1+x+1), se recorre el árbol, buscando los elementos únicos del árbol, ya sea números, variables, u operaciones, donde se agrupan en distintos sub-árboles de sumas, cada sub-árbol es evaluado, y finalmente se conectan todos en el árbol final.

Aspectos de implementación y eficiencia

La eficiencia no fue un objetivo de este laboratorio, por lo que no se buscó maneras de disminuir el tiempo de ejecución.

Debido a motivos de tiempo, la función *"expand()"* no pudo ser implementada, por lo que se agregó una 6ta y 7ma opción al menú, donde se permite operar la función *"sum_tree()"* y *"eval()"* individualmente, en caso de que estas entreguen una peor simplificación. Si la funcionalidad de expandir estuviera disponible, no habría problema, ya que esta se encargaría de expandir todos los elementos antes de generar el árbol de sumas. (A pesar de que la opción 2 es suficiente para la mayoría de los casos, de esta forma se puede observar fácilmente qué hace cada una)

La simplificación del árbol de sumas NO toma en cuenta el operador "-", si este nodo se encuentra en el árbol, es muy probable que no se genere una simplificación correcta.

La implementación de los casos de simplificación creó varias funciones demasiado largas y dificiles de leer, debido a que algunos casos de simplificación requerían de 2 a 4 casos distintos donde cambian las posiciones de los elementos. Estas secciones de código no podían ser generalizadas para los múltiples casos de una manera simple.

Casos de simplificación implementados: (incluidos ejemplos en los archivos de entrada)

- a - a = 0
- a - 0 = a
- a * 0 = 0
- a * 1 = a
- 0 ^ a = 0
- a ^ 1 = a
- a ^ 0 = 1
- a + 0 = a
- a + a = 2a
- a + n*a = n+1 * a
- na + ma = (m+n)a
- 1+x+1+x+1+x+1+x+1
- 1+x+1+y+1+x+1
- 1+x+1-x
- 1+x^2+1+x^2
- 1+2*x+1+3*x

Ejecución del código

En el proyecto se utilizaron las librerías *iostream*, *iomanip*, *fstream, stack*, *string* y *sstream* para obtener datos por consola, el nombre del archivo a leer, realizar la lectura del archivo y crear el árbol inicial, estas librerías fueron incluídas por el profesor, en el código que nos entregó de base. Luego, se incluyó la librería *vector*, para almacenar las operaciones en un arreglo, y aunque se pretendía usar la función *sort()* incluída, se terminó implementando una función propia, ya que se requería ordenar el arreglo según un criterio propio.

Para ejecutar el código, se descomprime el archivo zip, ya sea usando click derecho y la opción de extraer, o con el comando 'unzip LucasMesias212666599', luego se abre una terminal dentro de la carpeta recién extraída y se usa el comando 'make'. Se compilarán todos los archivos, y se podrá ejecutar cualquiera de los siguientes programas:

- ./main
- ./auto_txts (ejecuta las operaciones 2)
- ./testNodeNumber
- ./testNodeOperation
- ./testNodeVariable
- ./testloadFile (principalmente código entregado por el profesor)

Para ejecutar la prueba principal, los archivos de lectura estar en la carpeta *"txts"* que contiene el código fuente. (Todos los programas se ejecutan desde la carpeta principal del código fuente, los que leen texto desde una sub-carpeta acceden a ella por sí mismos)