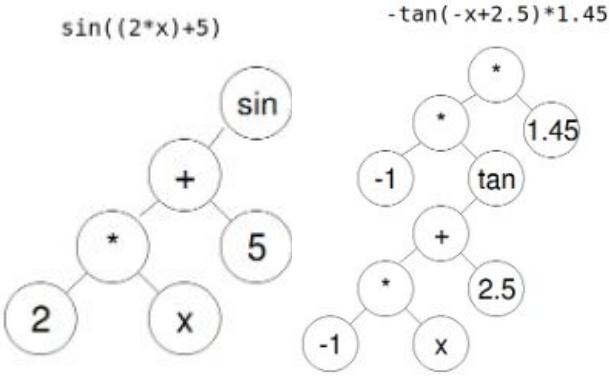
ALBRECHT Jérémy CB2
DELFORGE Bastien CB2

IMT Lille Douai
École Mines-Télécom
IMT-Université de Lille

Rapport TB7 : Analyse syntaxique

Cette seconde partie du projet consiste à traduire la liste chaînée créée grâce à l'analyseur lexical en un arbre binaire. Cet arbre va être utilisé par Louise et Flavie afin de calculer l'image d'un nombre par une fonction. De plus, l'arborescence permet de rendre compte des priorités d'opération (cf schémas ci-contre).



Avant de créer l'arbre, nous avons implémenté des règles pour que l'expression soit retranscrite dans l'arbre correctement :

- On ne peut pas rentrer ")" directement après "(".
- Il faut qu'il y ait le même nombre de "(" que de ")" dans l'expression.
- Après une fonction (cos, sin, tanh, exp, ln, SQRT) il faut qu'il y ait une parenthèse ouvrante, une expression et une parenthèse fermante.
- Il faut entrer une expression dans une fonction.
- On ne peut pas mettre 2 opérateurs à la suite et pas d'opérateur en début (sauf "-" et "+") et fin d'expression.

Si la fonction syntaxChecker() détecte une règle non respectée, elle retourne le type d'erreur (SYNTAX_ERROR ou PAR_ERROR généralement) et dans le cas contraire elle retourne NO_ERROR. S'il n'y a pas d'erreur, on peut appeler la fonction createTree() qui va créer et compléter l'arbre à partir de la liste chaînée. Pour implémenter cette fonction, nous avons adopté la stratégie suivante :
Nous commencons par chercher l'opérateur principal, c'est-à-dire celui qui est le moins encapsulé dans les parenthèses. Pour cela, nous parcourons la chaîne en incrémentant un compteur à chaque fois que l'on rencontre "(" et nous le décrémentons lorsque nous rencontrons ")". Ce compteur permet de savoir à quel niveau d'encapsulation dans les parenthèse un opérateur se trouve et ainsi de trouver la position de l'opérateur principal dans le fonction.

Arrivé ici, il y a 2 cas : il y a un opérateur principal dans la fonction ou il n'y en a pas (ex: f(x)= cos(x)).

Cas 1 : on crée l'arbre en mettant l'opérateur en noeud principal, on crée 2 listes chaînées : une qui contient tout ce qu'il y a avant l'opérateur et une qui contient tout ce qu'il y a après. En fonction du niveau d'encapsulation de l'opérateur principal, on ajuste le nombre de parenthèses ouvrantes et fermantes de nos 2 nouvelles listes. On finit par appeler de manière récursive la fonction createTree() pour chacun des fils. Pour le fils gauche on met en paramètre la première liste chaînée et pour le fils droit la seconde liste chaînée.

Cas 2 : dans ce cas comme il n'y a pas d'opérateur principal, nous avons fait un "switch" afin de déterminer la nature du premier maillon de la chaîne (variable, réel, fonction ou opérateur non principal). Si c'est une variable ou un nombre réel, c'est le cas d'arrêt de la fonction récursive, c'est-à-dire qu'on place l'élément comme noeud de l'arbre sans faire de nouvel appel. Si c'est une fonction, nous mettons cette fonction dans le noeud de l'arbre et nous nous sommes accordés avec Louise et Flavie pour mettre le contenu de la fonction dans le fils gauche. Finalement si c'est un opérateur non principal (ex: f(x) = -2*x), nous avons fait le choix de mettre un opérateur " * " dans le noeud de l'arbre, de mettre -1 dans un des fils et nous mettons le reste de la liste chaînée dans l'autre, ce qui revient à multiplier l'expression par -1. (Nous aurions également pu mettre " - " dans le noeud de l'arbre et mettre 0 dans le fils gauche et le reste de l'expression dans le fils droit, ce qui revient à faire "0 - expression").

Problèmes rencontrés : Identifier la position de l'opérateur principal pour ensuite créer les listes chaînées. En effet, la gestion des parenthèses nous a posé un soucis. Lorsque l'expression commençait par une parenthèse, nous avons fait en sorte que le programme la retire mais nous avions oublié de modifier l'indice qui indiquait la position de l'opérateur principal. Ainsi, nos 2 listes chaînés ne correspondaient pas à ce que nous attendions. Ensuite, lorsque l'on rentrait beaucoup de parenthèses comme f(x) = (((2*x))), le programme ne retirait que la première, ce qui posait aussi problème lors du traçage de la courbe mais nous avons corrigé le problème grâce à une boucle while.

Pistes d'amélioration : Le principal inconvénient réside dans l'utilisation des parenthèses et de la priorité des calculs. En effet, le polynôme x^2+x-2 doit être rentré sous la forme (x^2+x)-2 ou x^2+(x-2) pour que l'algorithme trouve l'opérateur principal. De même pour les expressions telles que 3*x-x, l'algorithme ne sait pas qu'il faut effectuer la multiplication d'abord si on ne met pas de parenthèse. Finalement, les expressions avec des multiplications implicites comme (3+x)(x-2) ne peuvent pas être retranscrites dans l'arbre.