Franck Caron, Pierre Dechamps
M1 GIL

Projet de Compilation

Introduction

Ce document a pour but de décrire synthétiquement les structures et grammaires employées, qui nous ont permis de produire ce convertisseur.

Ce dernier utilise l'analyseur lexical Lex, et l'analyseur grammatical YACC, pour pouvoir convertir une formule mathématique exprimée en langage LaTeX, en un code MathML utilisable pour l'affichage sur un site quelconque.

A cet effet, le programme se décompose en 3 modules principaux :

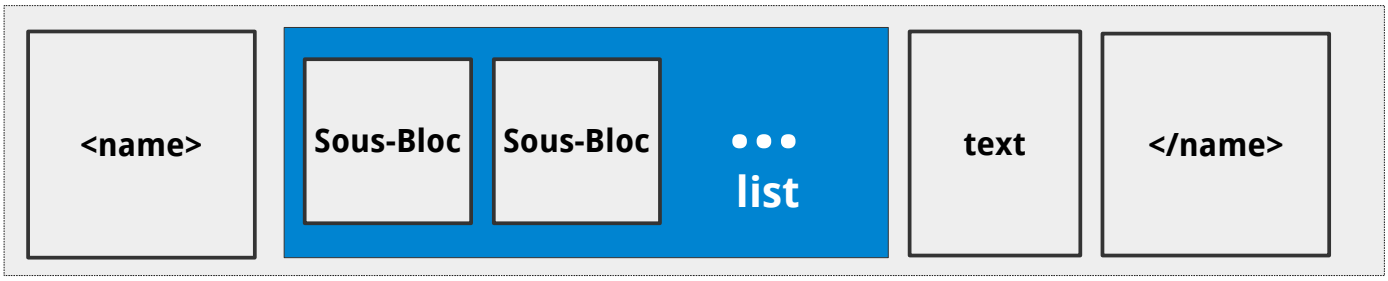
- Le module du modèle de structuration (bloc.c/bloc.h)
- Le module lexical (l2m.lex)
- Le module grammatical (l2m.y)

Structuration

Le principe du programme étant de fournir du code MathML (basé sur des balises HTML), il nous a semblé intéressant d'opter pour une structure qui permettrait de simuler une arborescence de blocs. De cette manière, il nous suffirait qu'à simplement imbriquer correctement des blocs afin de pouvoir produire le code final via un simple procédé récursif.

On définit ainsi le type bloc. (type bloc_t, cf bloc.h pour la structure interne)

Son contenu, une fois converti peut se représenter de la manière suivante :



Attributs

- name : Définit le nom de la balise qui sera utilisé par MathML pour afficher cet élément.
- List : La liste des blocs contenus par ce bloc, permet de créer le phénomène d'arborescence.
- Text : Le texte de fin de bloc, permet d'ajouter un contenu texte supplémentaire avant la balise de fermeture du bloc.

Cette structure assez souple permet d'être utilisable pour chacune des expressions LaTeX à implanter. On notera 3 types d'utilisations notables utilisées ici :

- Le conteneur : On ne définit que le nom et les sous-blocs, parfait pour la plupart des expressions complexes (frac, sum, sqrt, ...)
- Le contenu : On ne définit que le nom et le texte, ce qui nous permet de modéliser tous les éléments terminaux (balises mi, mn et mo).
- Le bloc vide : Seule la liste de sous-blocs est définie, ce qui simule la concaténation des chaînes représentant plusieurs blocs.

Ainsi, on définit les différentes primitives qui nous seront nécessaires pour l'exploitation de la formule MathML :

- **new_bloc :** Crée un nouveau bloc avec le nom et le texte fournis en paramètres. Ces paramètres sont optionnels, et ne seront pas pris en compte s'il ne sont pas définis.
- **empty_bloc :** Crée un nouveau bloc vide.
- **add_bloc :** Rajoute le 1er bloc, dans le contenu du 2è bloc, permet de construire l'arborescence des blocs.
- **consume_bloc :** Détruit le bloc tout en construisant la chaîne de caractères représentant le code MathML produit.

En exploitant le type de retour des expressions grammaticales et en gérant correctement les actions associées aux règles, cette structure permettra de construire simplement le code associé à la formule.

Traitement

Tout d'abord, on redéfinit le type YYSTYPE, afin de pouvoir manipuler des blocs lors du traitement des expressions (type bloc_t *).

On définit ensuite dans la partie C, des fonction utilitaires qui permettront des simplifier la syntaxe des actions.
Le principe est de définir une fonction de la forme new_blocX(args), qui va créer un bloc de nom X, avec comme sous-blocs, ceux entrés en argument.

L'ensemble des terminaux est alors défini dans le fichier l2m.lex, avec une condition particulière : Si le terminal correspond à un opérateur, un nom de variable, ou bien un nombre, nous allons alors retourner par la variable yylval, le bloc correspondant à son contenu (avec le bon nom de balise).

La construction de l'arborescence se fait alors au fur et à mesure des actions engendrées par les règles grammaticales de l2m.y.
A la fin de l'analyse, le bloc contenant l'intégralité de l'arborescence sera consumé, et le code MathML sera alors renvoyé sur la sortie standard.

Utilisation

Afin d'utiliser l'analyseur, vous devez tout d'abord le construire à l'aide de la commande make. Pour cela, il suffit d'ouvrir un terminal dans le répertoire du projet et d'y entrer la commande.

Une fois construit, vous pourrez utiliser l'analyseur au moyen de la commande **./l2m « formule »**, où formule représente la formule mathématique à convertir sans les caractères « $ » de début et fin de formule.