POLYTECH LILLE

Université de Lille

RAPPORT DE PROJET

Projet Langage et Traducteur - Semestre 6

Rémy Gaudré – Salohy Ramananjaraniaina
GIS 3A

Table des matières

Présentation du projet

L'objectif fixé lors de ce projet est la production d'un analyseur lexical et syntaxique du langage de mise en page LATEX. Le traducteur produit reconnaîtra la syntaxe de déclaration des documents LATEX qui après compilation permettra d'obtenir un document HTML.

Analyse du projet

Ce projet comporte trois étapes qui sont illustrés par les trois versions dans ce rapport. La première étape étant la définition de la grammaire, la seconde correspond à la vérification de la cohérence et la dernière à la génération du code HTML.

Version 1 : Définition de la grammaire

1. Reconnaissance des commandes Latex

Syntaxiquement, un document LATEX peut être considéré comme une suite de mots et de commandes. Un mot peut contenir des minuscules, majuscules, chiffres, caractères accentués et caractères parmi l'ensemble { : , ; . - ' ( )}.

Nous avons donc défini une lettre de mot ainsi :

```
lettre [0-9a-zA-Z:;,.\-'\(\)éèàùîê]
```

et le séparateur ainsi :

```
separateur [ \t]
```

Ensuite, nous avons cherché à reconnaître les commandes Latex. Toutes les commandes Latex commencent par un '\'. On peut distinguer cependant trois catégories de commandes :

1. Les commandes simples sans paramètres. Le caractère '/' est suivi du nom de la commande.
   Syntaxe : `\nomCommande`
2. Les commandes dont l'action s'applique dans une zone définie par des accolades.
   Syntaxe : `\nomCommande{zone…}`
3. Les commandes dont l'action s'applique dans une zone définie par des commandes de début et de fin.
   Syntaxe : `\begin{nomCommande} zone… \end{nomCommande}`

La liste des commandes que nous devons reconnaître est la suivante :

Nom Commande	Catégorie	Désignation
Document	3	Délimite un document Latex
Title	2	Définit le titre du document
Author	2	Définit l'auteur du document
Date	2	Définit la date du document
Maketitle	1	Génère un titre
Section	2	Définit un titre de niveau 1
Subsection	2	Définit un titre de niveau 2
Subsubsection	2	Définit un titre de niveau 3
Textbf	2	Définit une zone en gras
Textit	2	Définit une zone en italique
Itemize	3	Définit une zone de liste non numérotée
Enumerate	3	Définit une zone de liste numérotée
Item	1	Définit un élément d'une liste
\	1	Passage à une nouvelle ligne
Ldots	1	Points de suspension

Nous avons alors défini des macros (voir annexe 1) pour reconnaître les terminaux suivants :

- PARFER
- DEBUTDOC
- FINDOC
- TITREDOC_DEB
- AUTEURDOC_DEB
- DATEDOC_DEB
- TITRE
- TITRE1_DEB
- TITRE2_DEB
- TITRE3_DEB
- GRAS
- ITALIQUE
- LISTENONUM_DEB
- LISTENONNUM_FIN
- LISTENUM_DEB
- LISTENUM_FIN
- ITEM
- ANTISLASH
- SUSPENSION
- Mot

2. Définition de la grammaire

Afin de reconnaître les documents Latex, nous avons créé les non-terminaux suivants :

- **Document** : l'ensemble du document Latex y compris les commandes optionnelles
- **Opts** : Ensembles des commandes optionnelles
- **Opt** : Choix des commandes optionnelles possibles : **title**, **author**, **date**
- **DOC** : délimite un document Latex
- **TEXTE** : lit les mots du document
- **EXP** : expression en italique et en gras
- **Liste** : défini une liste ordonnée ou une liste non-ordonnée
- **Liste_items** : défini les éléments d'une liste
- **CONTENU** : génère un titre et des sections s'il y en a.
- **SECTION** : génère une section **TITRE1** suivi s'il y en a de sous-section **TITRE2** suivi lui-même de **sous-sous-sections TITRE3**

Le fichier projet.acc (voir annexe 2) contient cette grammaire.

3. Tests effectués

Après avoir compilé, nous avons pu tester le fichier contenu.tex (voir annexe 4). Nous avons vérifié que chaque instruction était bien reconnue par les macros et que la grammaire fonctionnait.

Version 2 : Vérification de la cohérence

1. Analyse des besoins de la grammaire

Les options de documents

Nous savons que le titre, l'auteur et la date ne peuvent être déclaré qu'une seule fois dans un document. Nous devons donc créer une contrainte. Voici notre solution :

On créée 3 variables synthétisées qui compterons le nombre d'auteur, de date et de titre de document définit :

- OptionTitre : Compte le nombre de titre
- OptionAuteur : Compte le nombre d'auteur
- OptionDate : Compte le nombre de date

A chaque fois qu'une nouvelle option est définie, on additionne le nombre de titre, de date et d'auteur défini. Ce qui donne, par exemple, l'arbre d'analyse suivant s'il y a un titre et une date défini à la suite dans le document (à noter que l'arbre n'est absolument pas complet, c'est juste une illustration de notre solution) :

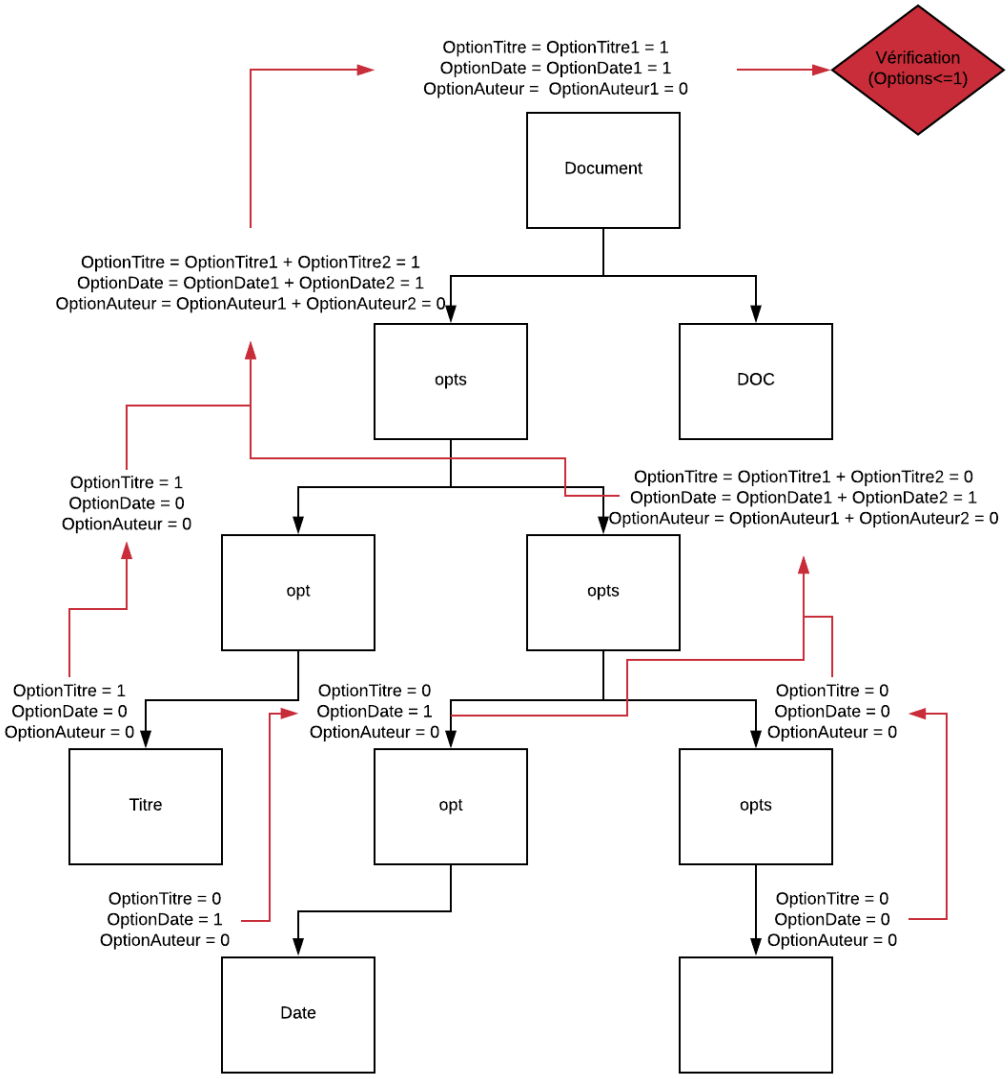

Figure 1 Exemple d'abre d'analyse suivant la déclaration d'un titre et d'une date dans un document Latex

Les sections

La seconde contrainte que nous devons prendre en compte est la déclaration des sous-sections et sous-sous-sections. Une sous-section ne peut pas être déclarée s'il n'existe pas déjà une section déclarée. Et une sous-sous-section ne peut pas être déclarée s'il n'y a pas encore de sous-section déclarée.

Pour répondre à ce problème, nous avons pensé à des variables hérités :

- Niv1 : Sert à définir si une section a été déclarée.
- Niv2 : Sert à définir si une sous-section a été déclarée.

A chaque fois qu'une section est déclarée, on met la variable niv1 à 1. Si jamais une sous-section est déclarée, on vérifie que cette variable est bien à 1. Et pareil avec la sous-section : si une sous-section est déclaré, on met la variable niv2 à 1. Et si une sous-sous-section est déclarée, on vérifie que la variable niv2 est bien à 1. On peut illustrer notre solution par l'arbre d'analyse suivant pour une section, une sous-section déclarée (à noter que l'arbre n'est absolument pas complet, il sert juste à illustrer notre solution) :

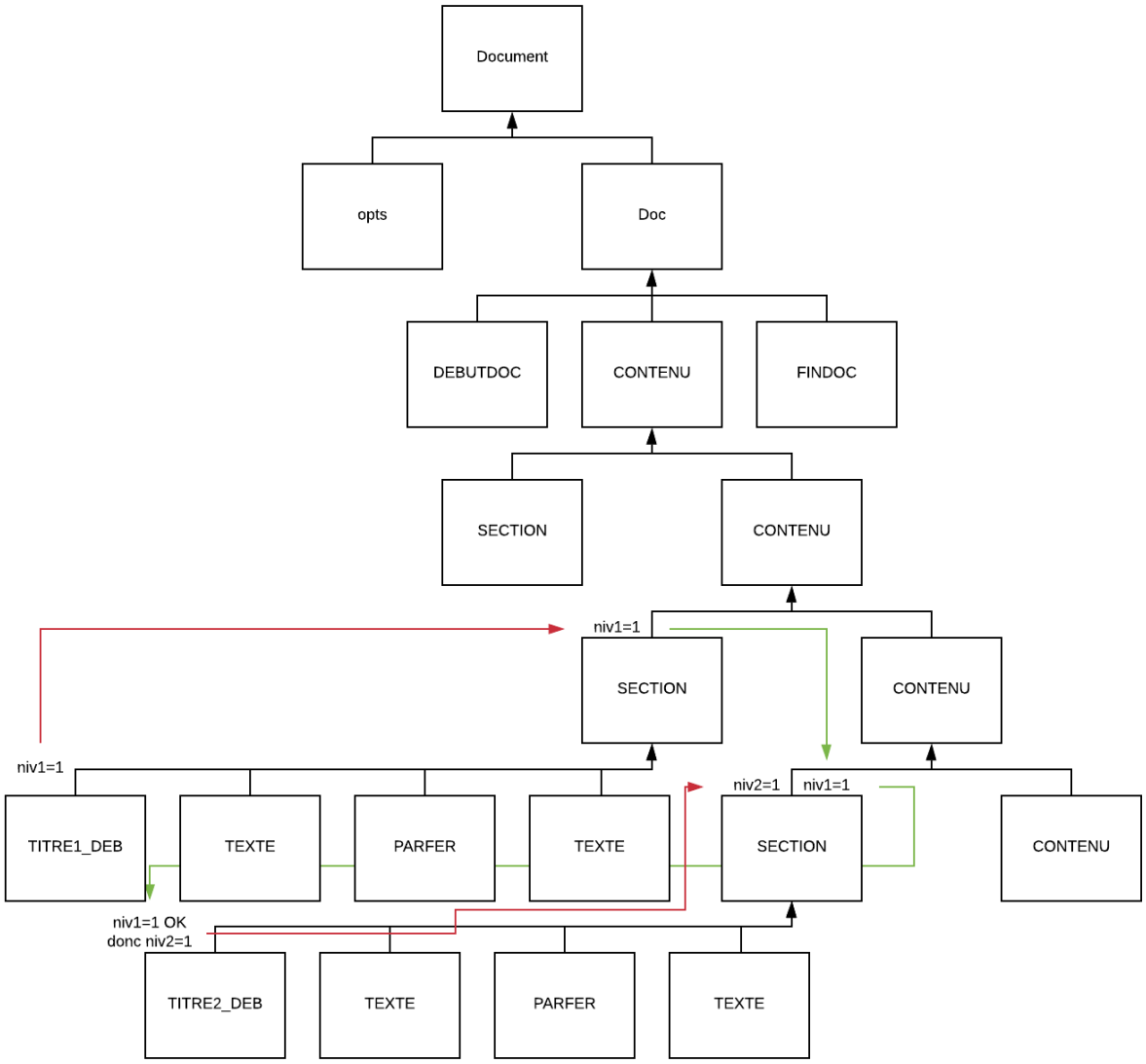

Figure 2 Exemple d'arbre d'analyse de la déclaration d'une section et d'une sous-section

2. Tests effectués

Nous avons testé toutes les possibilités suivantes pour les options :

- Si les options (date, titre et auteur) sont défini plus d'une fois
- Si on peut ne pas déclarer les options
- Si on peut en déclarer certaines une fois
- Si on peut en déclarer certaines plusieurs fois et d'autre non
- Si on peut toute les définir une fois

Nous avons aussi testé toutes les possibilités suivantes pour la déclaration des sections :

- Commande subsection sans section
- Commande subsubsection sans subsection
- Commande subsection avant section
- Plusieurs commandes subsection suivis de plusieurs commandes subsubsection

Nous avons eu tous les résultats attendus durant les tests

Version 3 : Génération HTML

1. Gestion du texte du document

Afin de pouvoir récupérer le texte du document, nous avons déclaré (voir fichier yystype.h en annexe 3):

- Texte : un tableau de caractère
- Attribute : une structure ayant comme unique champ texte
- YYSTYPE : une constante de type attribute pour stoker les chaînes de caractères reconnus dans le document source.

2. Impression des balises correspondantes au langage HTML

Suivant les commandes Latex reconnus dans la grammaire, nous avons ajouté des instructions C pour écrire les balises HTML dans le fichier de destination (voir annexe 2). En effet des commandes reconnues dans le document LATEX peuvent être associé à des balises du langage HTML comme suit :

COMMANDES LATEX	BALISES HTML
DEBUTDOC	<body>
FINDOC	</body>
ITALIQUE	<i>
GRAS	<b>
TITRE1_DEB	<h1>
TITRE2_DEB	<h2>
TITRE3_DEB	<h3>
LISTENONUM_DEB	
LISTENONNUM_FIN	
LISTENUM_DEB	
LISTENUM_FIN	

3. Tests effectués

Nous avons réussi à reproduire l'exemple donné dans le sujet. Le fichier test contenu.tex (voir annexe 4) est parfaitement traduit en HTML.

Conclusion

A travers ce projet, nous avons pu illustrer et appliquer les notions abordées tout au long du cours de Langages et Traducteurs. Le but principal fut de concevoir un traducteur qui reconnaît la syntaxe de déclaration des documents LATEX. Cette tâche nous a permis de comprendre la partie frontale des phases d'un compilateur. Nous avons assimilé comment d'un flot de caractère donné en entrée, un analyseur lexical (construit par LEX) reconnaît et retourne à l'analyseur syntaxique (construit par ACCENT) un flot d'unités lexicales pour que ce dernier puisse ensuite vérifier la grammaire du flot d'entrée.

Pour des études encore plus poussées, un analyseur sémantique pourrait être produit en guise d'amélioration de ce traducteur, en plus d'ajouter d'autres fonctionnalités du langage html (comme la gestion des tableaux).

ANNEXES

1. Fichier projet.lex

```
%{
/* Fichier projet.lex */
/* Section en C : Declarations de constantes et de variables
                  Insertions de bibliotheques */
#include "yystype.h"
#include "yygrammar.h"
char err[40];

%}

/* Definition de macros */
lettre [0-9a-zA-Z:;,.\-'\(\)éèàùîê]
separateur [ \t]

%%
\}                          {
/*                                 printf("Fin"); */
                            return PARFER ;
                            }
"\\begin\{document\}"       {
                                return DEBUTDOC;
                            } /* Début du document */
"\\end\{document\}"         {
                                return FINDOC;
                            } /* Fin du document */
"\\title\{"                 {
                                return TITREDOC_DEB;
                            } /*Définition du titre */
"\\author\{"                 {*/
                                return AUTEURDOC_DEB;
                            } /* Définit l'auteur du document */
"\\date\{"                  {*/
                                return DATEDOC_DEB;
                            } /* Définit la date du document */
"\\maketitle"               {
                                return TITRE;
                            } /* Génère un titre */
"\\section\{"               {
                                return TITRE1_DEB;
                            } /* Définit un titre de niveau 1 */
"\\subsection\{"            {
                                return TITRE2_DEB;
                            } /* Définit un titre de niveau 2 */
"\\subsubsection\{"         {
                                return TITRE3_DEB;
                            } /* Définit un titre de niveau 3 */
"\\textbf\{"                {
                                return GRAS;
                            } /* Définit une zone de gras */
"\\textit\{"                {
                                return ITALIQUE;
                            } /* Définit une zone en italique */
"\\begin\{itemize\}"        {
                                return LISTENONNUM_DEB;
```

```
                            } /* Définit une zone de liste non numéroté
*/
"\\end\{itemize\}"          {
                                return LISTENONNUM_FIN;
                            } /*Finit une zone de liste non numéroté */
"\\begin\{enumerate\}"      {
                                return LISTENUM_DEB;
                            } /* Définit une zone de liste numéroté */
"\\end\{enumerate\}"        {
                                return LISTENUM_FIN;
                            } /*  Définit une zone de liste numéroté */
"\\item"                    {
                                return ITEM;
                            } /* Définit un élement d'une liste */
"\\"                        {
                                return ANTISLASH;
                            } /* Passage à une nouvelle ligne */
"\\ldots"                   {
                                return SUSPENSION;
                            } /* Point de suspension */
{lettre}+                  {  yylval.texte= (char *)
malloc(strlen(yytext+1)) ; strcpy(yylval.texte,yytext) ;
                                return mot;
                            }
{separateur}+               ;
\n                          yypos++ ;

.                           {sprintf(err,"Mauvais caractere
%c",yytext[0]);
                            yyerror(err);
                            }
%%
```

2. Fichier projet.acc

```
/* Fichier projet.acc */

%prelude{ /* Code C */
   /* Inclusion de bibliotheques C */
   #include<stdio.h>
   #include<malloc.h>
   #include <string.h>
#include "yystype.h"



   /* Action de fin d analyse */
   void fin_analyse(){
//         printf("Fin de traitement \n") ;
//         printf("Syntaxe correcte \n") ;
   }

   void verificationOptions(int OptionTitre, int OptionDate, int
OptionAuteur)
   {
        if(OptionTitre>1) yyerror("ERREUR : Le titre du document a été
défini plus d'une fois.\n");
        else if (OptionAuteur>1) yyerror("ERREUR : L'auteur du document
a été défini plus d'une fois.\n");
        else if (OptionDate>1)yyerror("ERREUR : La date du document a
été défini plus d'une fois.\n");
        else fin_analyse();
    }

}

/* Declaration des tokens */
%token DEBUTDOC, FINDOC, TITREDOC_DEB, AUTEURDOC_DEB, DATEDOC_DEB,
PARFER, TITRE, TITRE1_DEB, TITRE2_DEB, TITRE3_DEB, GRAS, ITALIQUE,
LISTENONNUM_DEB, LISTENONNUM_FIN, LISTENUM_DEB, LISTENUM_FIN, ITEM,
ANTISLASH, SUSPENSION, mot, SEPARATEUR ;

// Grammaire


Document : {printf("<!doctype html>\n<html>\n   <head>\n    <meta
charset='UTF-8'/>\n       <title>");} opts <OptionTitre,
OptionDate,OptionAuteur> {printf("</title>\n   </head>\n");}DOC
{printf("</html>");} {verificationOptions(OptionTitre, OptionDate,
OptionAuteur);}
;

opts <%out int OptionTitre, int OptionDate, int OptionAuteur> : opt
<OptionTitre1, OptionDate1, OptionAuteur1> opts <OptionTitre2,
OptionDate2, OptionAuteur2>
{*OptionTitre = OptionTitre1 + OptionTitre2;
    *OptionDate = OptionDate1 + OptionDate2;
    *OptionAuteur = OptionAuteur1 + OptionAuteur2;
}
     | {*OptionTitre = 0;*OptionDate = 0;*OptionAuteur = 0;}
     ;
```

```
opt <%out int OptionTitre, int OptionDate, int OptionAuteur > :
{*OptionTitre = 1;*OptionDate = 0;*OptionAuteur = 0;} TITREDOC_DEB
TEXTE PARFER
            | DATEDOC_DEB TEXTE PARFER {*OptionTitre = 0;*OptionDate =
1;*OptionAuteur = 0;}
            | AUTEURDOC_DEB TEXTE PARFER {*OptionTitre = 0;*OptionDate
= 0;*OptionAuteur = 1;}
;

DOC : DEBUTDOC {printf("    <body>\n");} CONTENU FINDOC
{printf("</body>\n");}
;

TEXTE : ANTISLASH {printf("\n");} TEXTE
        | EXP TEXTE
        | SUSPENSION {printf("...");} TEXTE
        | mot <ch> {printf(" %s",ch.texte);} TEXTE
        | liste TEXTE
        |
;

EXP : ITALIQUE {printf("<i>");} TEXTE PARFER {printf("</i>");} TEXTE
        | GRAS {printf("<b>");} TEXTE PARFER {printf("</b>");} TEXTE
;

liste : LISTENONNUM_DEB {printf("<ul>");} liste_items LISTENONNUM_FIN
{printf("</ul>\n");}
        | LISTENUM_DEB {printf("<ol>");} liste_items LISTENUM_FIN
{printf("</ol>\n");}
;

liste_items : ITEM {printf("<li>");} TEXTE {printf("</li>\n");}
liste_items
        |
;

CONTENU : TITRE CONTENU
        | SECTION <niv1,niv2> CONTENU
        |
;

SECTION <%in int niv1, int niv2 >: TITRE1_DEB {printf("<h1>");} TEXTE
PARFER {printf("</h1>\n");} TEXTE {niv1 = 1;}
        | TITRE2_DEB {printf("<h2>");} TEXTE PARFER
{printf("</h2>\n");} TEXTE {if (niv1 == 1){niv2 =
1;}else{yyerror("ERREUR la section de niveau 1 n'a pas été
définie.\n");}}
        | TITRE3_DEB {printf("<h3>");} TEXTE PARFER
{printf("</h3>\n");} TEXTE {if (niv2 == 1){}else{yyerror("ERREUR la
section de niveau 2 n'a pas été définie.\n");}}
;
```

3. Fichier yystype.h

```
/* fichier yystype.h*/
typedef union { char* texte; } attribute ;
#define YYSTYPE attribute
```

4. Ficher contenu.tex

```
\title{Un document latex}
\author{\textit{Polytech}}
\date{}

\begin{document}
\maketitle

\section{Document \textit{Latex}}

Un document \textbf{Latex} est un document texte qui contient certaines
commandes qui seront traitées par un compilateur. \\
L'une des forces de Latex est de pouvoir générer des documents très
propres, \textit{très structurés} avec une numérotation automatique.

\subsection{Fonctionnement}

En résumé, Latex :

\begin{itemize}
\item permet d'écrire des documents à l'aide d'un simple éditeur de
texte,
\item s'occupe de la mise en page,
\item offre un résultat \textit{incomparable}.
\item La réalisation d'un document se fait en 3 phases :
    \begin{enumerate}
    \item Edition du texte du document (vi, kwrite, xemacs, \ldots)
    \item Compilation du texte (latex, pdflatex)
    \item Visualisation du document généré (evince, acroread)
    \end{enumerate}
\end{itemize}

\end{document}
```