
Université
Paul Sabatier
TOULOUSE III

Analyse Syntaxique

Mohammed Akram Rhafrane - Ismail Senhaji - Nathanael Bertrand - Mehdi Boutchiche

11 avril 2013

Table des matières

Résumé

Résumé du contenu du document.

1 Introduction

2 Fondamenteux

2.1 Langages et grammaires

2.1.1 Défintion

Un langage formel est un ensemble de mots constitués de symboles qui appartiennent à son alphabet. Un langage formel est décrit par une grammaire. De manière générale, une grammaire est définie par un quadruplet :

N : l'ensemble des non-terminaux utilisés pour décrire les règles de productions

X : l'ensemble des terminaux, c'est à dire les symboles ou encore l'alphabet

P : l'ensemble des règles de production

S : l'axiome, c'est un élément de N

Ainsi, la notation : G(L) = <N, X, P, S> décrit la grammaire G associée au langage L. Les grammaires sont analysées par des automates. Il existe plusieurs types d'analyses de grammaires qui font appels à plusieurs types d'automates. Dans le monde de la compilation l'analyseur syntaxique fait référence à l'algorithme qui met en oeuvre l'automate d'analyse d'une grammaire. Dans cet écrit, nous nous attarderons sur deux types d'analyseurs : les analyseurs de type LL et ceux de type L(AL)R.

2.1.2 Grammaires ambiguë

On dit qu'une grammaire est ambiguë lorsqu'on peut trouver deux arbres de dérivation différents pour le même mot. Les grammaires ambiguë pose un problème lors de la compilation, c'est pour ça qu'il est préférable de les transformer en grammaire non ambiguë si c'est possible.

2.2 Analyse lexicale

l'analyse lexicale consiste à découper une chaîne de caractère en unités lexicales ou lexèmes à la demande de l'analyseur lexicale. Il est définit par un ensembles d'expressions relationnelle qui exigent certaines séquences de caractère pour former les lexèmes.

2.2.1 Segmentation

La segmentation est le fait de séparer les différentes sections d'une chaînes de caractères, par exemple pour une phrase l'ordinateur la considère comme une chaîne de caractère et non pas une suite de mot, le rôle de la segmentation est donc de faire une séparation entre ces mots selon le caractère de séparation dans ce cas le caractère espace.

2.2.2 Unités lexicales

Une unité lexicale ou un lexème est une chaîne de caractère qui correspond à un symbole. à l'aide du processus de segmentation, on peut extraire à partir d'un flux de caractères entrant une suite d'unités lexicales, ensuite c'est l'analyseur lexicale qui traitent ces lexème et les rangent dans des catégories d'entités lexicales.

2.3 Analyse syntaxique

Le rôle de l'analyse syntaxique est de savoir si une phrase appartient à la syntaxe d'un langage. A partir du flot de lexèmes construits par l'analyse lexicale dans un premier temps, l'analyse syntaxique permet de générer un arbre de syntaxe abstraite. Cet arbre est construit à base d'un ensembles de règles définissant une grammaire formelle sur laquelle est basé le langage en question. l'analyse syntaxique permet plus particulièrement de détecter les erreurs de syntaxe en continuant tout de même l'analyse pour éviter les cycles de compilation/correction pour les développeurs. un analyseur syntaxique doit retracer le cheminement d'application des règles qui ont menées à l'axiome. Pour ça, il existe deux types d'analyse :

2.3.1 Analyse ascendante

Le principe de l'analyse ascendante est de partir de l'axiome en suivant les règles de production afin de retrouver le texte analysé. Ce type d'analyse procède en découpant le texte petit à petit jusqu'à retrouver les unité lexicale. L'analyse LL est un exemple d'analyse descendante.

2.3.2 Analyse descendante

L'analyse descendante d'une autre part, procède contrairement à l'analyse syntaxique en retrouvant le cheminement à partir du texte analysé. Ce type d'analyse essaye de regrouper les unité lexicale entre elles pour retrouver l'axiome. L'analyse LR est un exemple d'analyse ascendante.

2.4 LL(*)

Descente récursive ou prédicative : pour les grammaires simple ou le premier symbole terminal fournie des informations suffisantes pour choisir la règle de production. Pas possible si grammaire récursive à gauche. Besoin de factorisation à gauche lorsque 2 règles commencent par le même lexème. Mais cette méthode à une faiblesse, c'est qu'elle doit toujours prédire qu'elle règle utiliser.

2.5 L(AL)R

Dérivation à droite permet de rapporter le choix de la règle de production à utiliser. Elle commence du bas vers le haut. L'algorithme s'arrête quand tous les caractères ont été lus. La chaîne est accépté si la partie analysée se réduit à l'axiome.

3 Comparaison entre les outils

3.1 Yac/Bison/Cup

3.1.1 Yac - Yet Anather Compiler-Compiler

YACC est un programme qui permet de générer un analyseur syntaxique à partir d'une spécification. La spécification est l'ensemble des règles de grammaire associées au langage à analyser. L'analyseur ainsi généré est de type LALR(1).

3.1.2 Specification

La spécification est l'ensemble des données qui permettront à YACC de générer l'analyseur. Elle décrit le langage qui sera reconnu sous forme de règles de grammaires. De cette façon l'analyseur a les connaissances pour définir si un flux donné en entrée est syntaxiquement correcte par rapport à sa spécification. Cependant, un analyseur syntaxique est souvent utilisé dans le contexte de traduction de langage. La spécification permet cette fonctionnalité puisqu'il est possible de spécifier des actions associées aux règles de grammaire.

La spécification suit la structure suivante :

Bloc des déclarations

Règles de grammaire

Programme

Les sections "Bloc des déclarations" et "Programme" sont facultatives.

Une règle de grammaire est notée sous la forme :

A : B /* action pour cette regle */ ;

Ci-dessous un exemple de spécification Yacc qui génère une calculatrice très minimaliste.

Ici, on utilise Yacc sans Lex afin de comprendre le fonctionnement. Cependant, il faut définir à la main la fonction yylex(). Celle-ci renvoie un entier qui indique le type de token qui a été reconnu en entrée.

```
#include <ctype.h> #include <stdio.h> #include <stdlib.h>
ligne : commande " ligne | " { printf("Fin du programme") ; exit(0) ; } ;
commande : expr { printf("Resultat : %d", $1) ; } ;
expr : expr '+' terme
$$ = $1 + $3 ; } | terme ;
terme : terme '*' facteur { $$ = $1 * $3 ; } | facteur ;
facteur : '(' expr ')' { $$ = $2 ;
| CHIFFRE ;
int main(){ yyparse() ; }
int yyerror(char *s){ printf("%s ", s) ; }
int yylex()
int c ; c = getchar() ; if(isdigit(c)){ yylval = c-'0' ; return CHIFFRE ; } return c ; }
```

Pour faire fonctionner cet exemple, il faudra entrer les lignes de commande suivantes :

```
> bison exemple1.y
```

> gcc exemple1.tab.c -o exemple1
> ./exemple1

Dans cet exemple, on a défini 5 règles. À quelques unes de ces règles on a associé des actions afin d'effectuer les calculs sur les chiffres lus.

Ces actions sont des instructions C dans lesquelles on a accès à des variables spéciables. Ainsi les variables $n font référence à la valeur du n-ième élément de droite et $$ fait référence à l'élément de gauche.

3.2 ANTLR

ANTLR est un Générateur de Parseur public, il permet d'utiliser plusieurs langages et intégre la description Lexical/Syntaxique.

3.3 Xtext

Xtext est un framework pour le développement de langages de programmation et de DSL(Domain Specific programming language). DSL est un langage de programmation dont les spécifications sont à un domaine d'applications précis , la construction des langages dédiés diffère fondamentalement de celle d'un langage classique, le processus de développement peut s'avérer très complexe, sa conception nécessite une double compétence sur le domaine à traiter et en développement informatique(exp :SQL :destiné à interroger ou manipuler une base de données relationnelle) Le framework Xtext s'appuie sur sur une grammaire générée ANTLR ainsi que le framework de modelisation EMF. Xtext couvre tous les aspects d'un IDE moderne : parseur, compilateur ,interpreteur et integration complete dans l'environnement de developpement Eclipse. Xtext fournit un environnement convivial aux utilisateurs d'un DSL. Parmi les fonctionnalités qu'offre Xtext : +coloration syntaxique ; suivant les éléments de la grammaire , Xtext propose une coloration syntaxique entièrement personnalisable. +auto complétion : auto complétion sur les éléments du langage +validation : Xtext valide le contenu de l'éditeur à la volée, produisant ainsi un retour direct à l'utilisateur en cas d'erreur de syntaxe. +intégration avec d'autres composants Eclipse

4 Conclusion