Grammaires et outils d'analyse de langages de programmation

SENHAJI Ismail
RHAFRANE Mohammed Akram
BOUTCHICHE Mehdi
BERTRAND Natnahel

29 mars 2013

Table des matières

1 Résumé **5**

2 Introduction **6**

3 Fondamenteux **7**
3.1 Langages et grammaires 7
3.1.1 Défintion 7
3.1.2 Grammaire ambigue 7
3.2 Analyse Lexicale 7
3.2.1 Défintion 7
3.2.2 Segmentation 8
3.2.3 Unités lexicales 8
3.3 Analyse syntaxique 8
3.3.1 Défintion 8
3.3.2 Analyse descendante 8
3.3.3 Analyse ascendante 8
3.4 LL * 9
3.4.1 Défintion 9
3.5 LL A R 9
3.5.1 LR 9
3.5.2 LALR 9

4 Comparaison entre les outils **10**
4.1 YACC/BISON/CUP/JFLEX 10
4.1.1 YACC 10
4.1.2 Spécifications 10
4.2 ANTLR 10
4.2.1 Défintions 10
4.2.2 Features 11
4.2.3 Elèments de langages 11
4.2.4 Grammaire 11
4.2.5 Gestion d'erreurs 11
4.2.6 L'arbre 11
4.3 XTEXT 11
4.3.1 Défintions 11

5 Conclusion **13**

6 Références **14**

7 Some small hints **15**
7.1 German Umlauts and other Language Specific Characters 15
7.2 References 15
7.3 Dividing Large Documents 15

List of Figures **16**

List of Tables **17**

Chapitre 1

Résumé

Chapitre 2

Introduction

Chapitre 3

Fondamenteux

3.1 Langages et grammaires

3.1.1 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.

3.1.2 Grammaire ambigue

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.

3.2 Analyse Lexicale

3.2.1 Défintion

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.

3.2.2 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.

3.2.3 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.

3.3 Analyse syntaxique

3.3.1 Défintion

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 :

3.3.2 Analyse descendante

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.

3.3.3 Analyse ascendante

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.

3.4 LL *

3.4.1 Défintion

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.

3.5 LL A R

3.5.1 LR

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.

3.5.2 LALR

Utilisé lorsque le traitement des données doit répondre à de multiples cas et lorsque la résolution par la programmation "standard" ne permettait pas une maintenance facile. Yacc et GNU Bison sont des analyseur grammaticaux.

Chapitre 4

Comparaison entre les outils

4.1 YACC/BISON/CUP/JFLEX

4.1.1 YACC

Utilisé lorsque le traitement des données doit répondre à de multiples cas et lorsque la résolution par la programmation “standard” ne permettait pas une maintenance facile. Yacc et GNU Bison sont des analyseur grammaticaux.

4.1.2 Spécifications

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.

4.2 ANTLR

4.2.1 Défintions

ANTLR est un Générateur de Parseur public, il permet d’utiliser plusieurs langages et intégre la description Lexical/Syntaxique. Grammaire LL k ANTLR est un PARSEUR qui permet d’effectuer toutes les tâches où les autres PARSEUR échoue, il permet aussi d’établir le lien entre le SCANNER, rapporter les erreurs et aussi construire l’arbre syntaxique, ceci sont des actions généralement fait à la main dans le cas d’autres PARSEUR.

4.2.2 Features

Différencier entre l'analyse Syntaxique et Lexical Faciliter de construction d'un arbre syntaxique Faciliter de détection d'erreur Portabilité en C . . ..

4.2.3 Elèments de langages

Grâce à des éléments de langages, ANTRLR différencie entre le Prédicat Syntaxique et Semantique comme l'exemple précèdent.

4.2.4 Grammaire

ANTLR à aussi besoin de beacoup mois de LL pour différencier entre les alternatives de fins exemple : LL3 pour un autre Parseur Vs LL 1 pour ANTLR ANTLR utilise des labels pour accéder au attributs , on utilisant ce procédé, la grammaire et beacoup plus lisible. PARSEUR classique ce référence à la table des symboles pour résoudre une ambiguïté syntaxique ANTLR Utilise un prédicat syntaxique grâce à ces éléments de langages. Une description ANTLR contient à la fois la spécification (Analyseur Lexical) et la grammaire ce qui permet de les grouper dans un seule fichier. Ceci est fait grâce au élément ANTLR (et les labels) qui permet de détecter et de différencier entre les grammaires et les TOKENS ANTLR permet aussi d'utiliser plusieurs Analyseur lexical dans la même description ANTLR

4.2.5 Gestion d'erreurs

Heuristique bien définie et efficace (gestion simple) PARSER EXCEPTION HANDLING pour une gestion plus sophistiquer

4.2.6 L'arbre

Grâce à ces éléments de langage, ANTLR réussi à construire automatiquement son AST ABSTRACT SYNTAXIC TREE ANTLR génère du code C et Cpp pour un Parseur de descente récursive ou toutes les règles de grammaires sont réalisé par des fonctions C / C++

4.3 XTEXT

4.3.1 Défintions

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

Chapitre 5

Conclusion

Chapitre 6

Références

Chapitre 7

Some small hints

7.1 German Umlauts and other Language Specific Characters

You can type german umlauts like 'ä', 'ö', or 'ü' directly in this file. This is also true for other language specific characters like 'é', 'è' etc. There are problems with automatic hyphenation when using language specific characters and OT1-encoded fonts. In this case, use a T1-encoded Type1-font like the Latin Modern font family (`\usepackage{lmodern}`).

7.2 References

Using the commands `\label{name}` and `\ref{name}` you are able to use references in your document. Advantage : You do not need to think about numerations, because LaTeX is doing that for you. For example, in section 7.3 on page 15 hints for dividing large documents are given. Certainly, references do also work for tables, figures, formulas... Please notice, that LaTeX usually needs more than one run (mostly 2) to resolve those references correctly.

7.3 Dividing Large Documents

You can divide your LaTeX-Document into an arbitrary number of TeX-Files to avoid too big and therefore unhandy files (e.g. one file for every chapter). For this, you insert in your main file (this one) for every subfile the command '`\input{subfile}`'. This leads to the same behavior as if the content of the subfile would be at the place of the `\input`-Command.

Table des figures

Liste des tableaux