Analyse Syntaxique

Mohammed Akram RHAFRANE - Mehdi BOUTCHICHE - Nathanael BERTRAND - Ismail SENHAJI

Université de Toulouse III/IRIT

Année 2012/2013

Plan

Objectif du TER

- Initiation à la recherche bibliographique
- Rédaction :
  Rapport sur le sujet (40 pages)
  Rapport du TER (10 pages)
  Présentation du TER
- Tous les livrables sont rédigés en Latex

Plan

Plan

Contexte

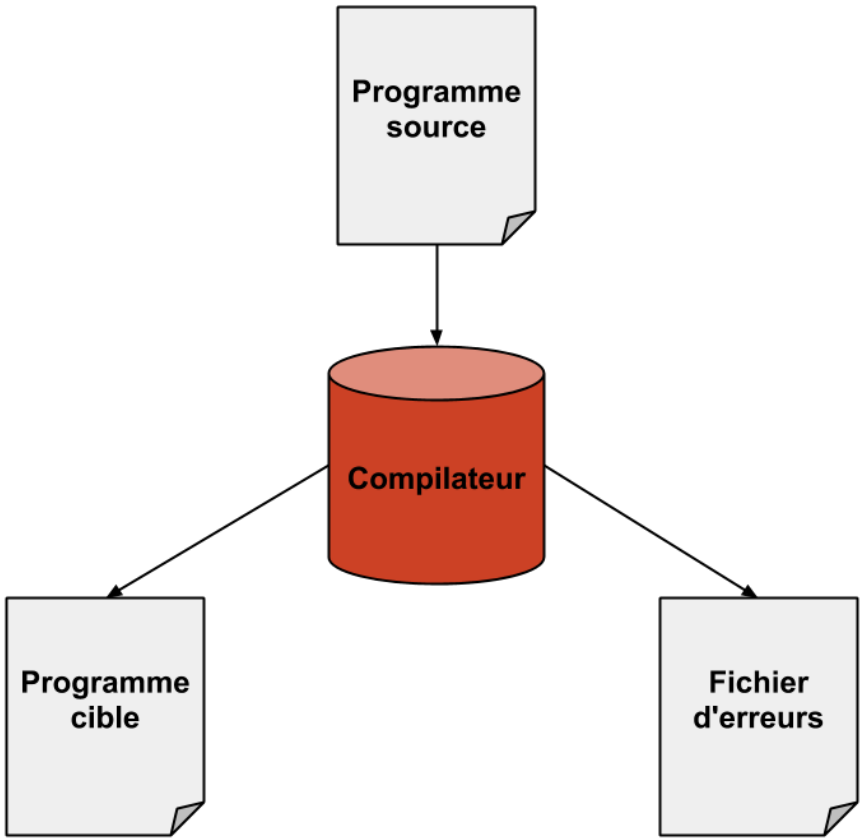


Figure : Compilateur

Contexte

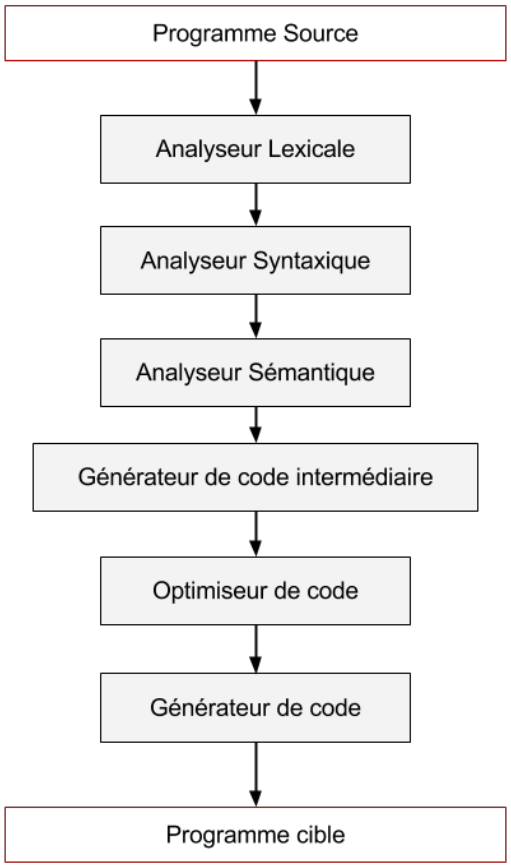


Figure : Processus de compilation

Plan

Analyse Lexicale
Principe

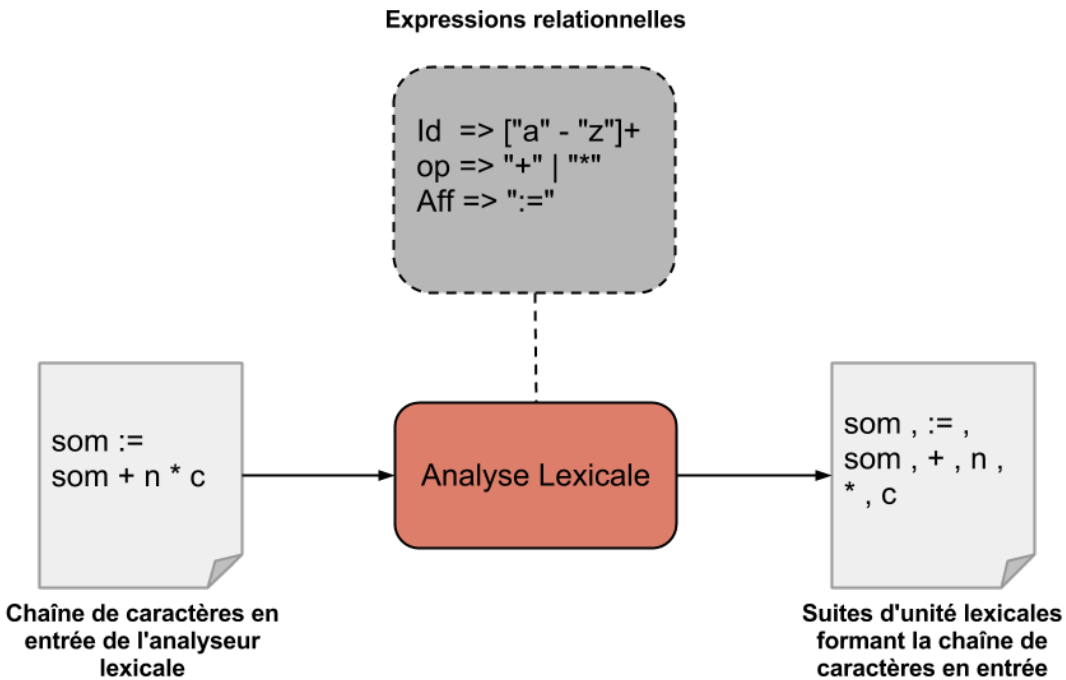


Figure : Analyse lexicale

Plan

Analyse Syntaxique
Principe

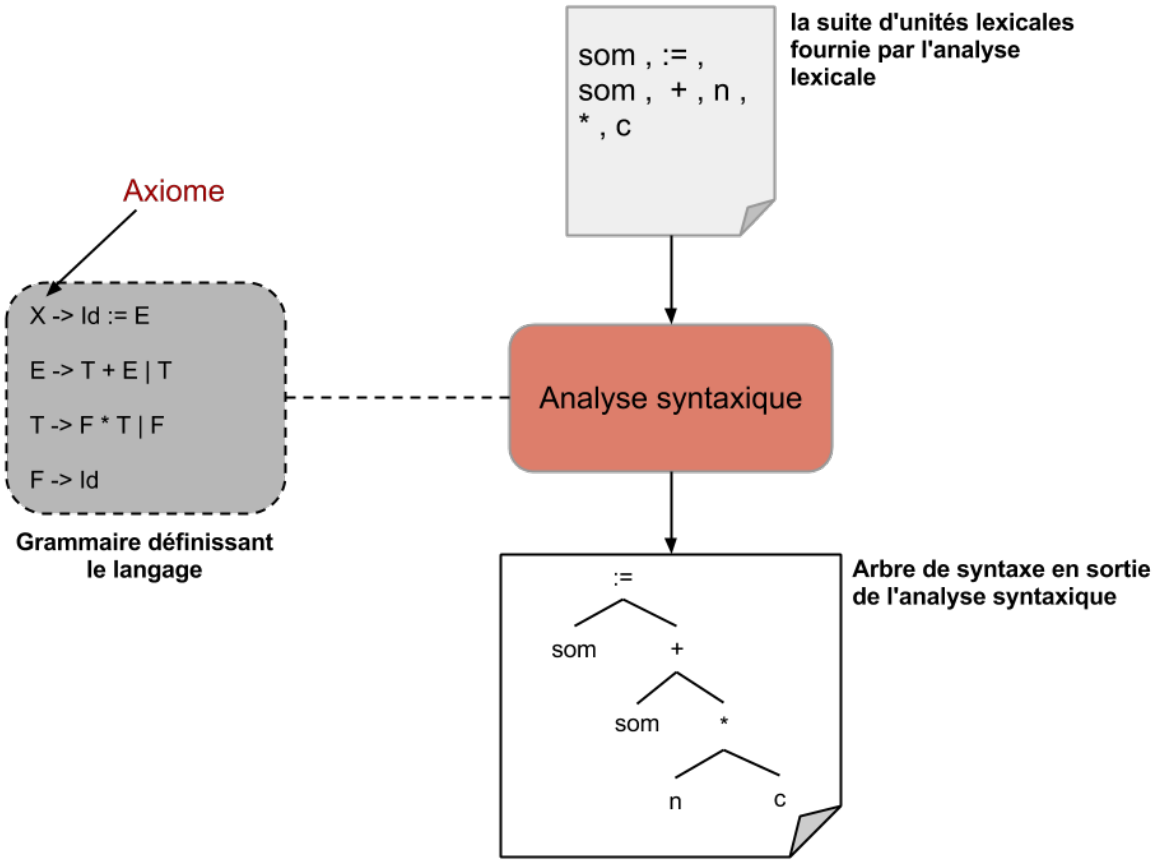


Figure : Analyse lexicale

Analyse Syntaxique
Exemple

S => S ; S
S => id := E
S => print(L)
E => id
E => num
E => E + E
E => (S , E)
L => E
L => L , E

id := num; id := id + (id := num + num, id)

Analyse Syntaxique
Exemple

$\underline{S}$
$S\ ;\ \underline{S}$
$\underline{S}\ ;\ id := E$
$id := \underline{E}\ ;\ id := E$
$id := num\ ;\ id := \underline{E}$
$id := num\ ;\ id := E + \underline{E}$
$id := num\ ;\ id := \underline{E} + (S\ ,\ E\ )$
$id := num\ ;\ id := id + (\underline{S}\ ,\ E\ )$
$id := num\ ;\ id := id + (id := \underline{E}\ ,\ E\ )$
$id := num\ ;\ id := id + (id := E + E\ ,\ \underline{E}\ )$
$id := num\ ;\ id := id + (id := \underline{E} + E\ ,\ id\ )$
$id := num\ ;\ id := id + (id := num + \underline{E}\ ,\ id\ )$
$id := num\ ;\ id := id + (id := num + num\ ,\ id\ )$

Plan

Analyse LL
Définition

- Left to right
- Rightmost derivation
- LL(k)

Analyse LL
Exemple

mot analysé: id := num ; id := id + (id := num + num, id)

$\underline{S}$

$\underline{S}$; S

id := $\underline{E}$; S

id := num ; $\underline{S}$

id := num ; id := $\underline{E}$

id := num ; id := $\underline{E}$ + E

id := num ; id := id + $\underline{E}$

id := num ; id := id + ($\underline{S}$, E)

id := num ; id := id + (id := $\underline{E}$, E)

id := num ; id := id + (id := $\underline{E}$ + E, E)

id := num ; id := id + (id := num + $\underline{E}$, E)

id := num ; id := id + (id := num + num, $\underline{E}$)

id := num ; id := id + (id := num + num, id)

Plan

Analyse LR
Définition

- Left to right
- Leftmost derivation
- LR(k)

Analyse LR
Exemple

mot analysé: id := num ; id := id + (id := num + num, id)

$\underline{S}$

S ; $\underline{S}$

S ; id := $\underline{E}$

S ; id := E + $\underline{E}$

S ; id := E + (S, $\underline{E}$)

S ; id := E + ($\underline{S}$, id)

S ; id := E + (id := $\underline{E}$, id)

S ; id := E + (id := E + $\underline{E}$, id)

S ; id := E + (id := $\underline{E}$ + num, id)

S ; id := $\underline{E}$ + (id := num + num, id)

$\underline{S}$; id := id + (id := num + num, id)

id := $\underline{E}$; id := id + (id := num + num, id)

id := num ; id := id + (id := num + num, id)

Analyse LR
Types d'analyseurs LR

- Analyseurs LR simples (SLR)
- Analyseurs syntaxiques LR avec anticipation (LALR)
- Analyseurs syntaxiques canoniques

Plan

Plan

YACC
Définitions

- Génération d'analyseur syntaxique
  A partir d'une spécification
- Langage : C
  Les analyseurs syntaxique généré par YACC sont en langage C
- Type LALR
  Les analyseurs syntaxique généré par YACC sont de type LALR

YACC
Spécifications

Les spécifications permettent:

- Description du langage
- Spécification des actions associées aux règles de grammaires
- Génération d'analyseur syntaxique

YACC
Structure de la spécifications

La spécification suit la structure suivante :
Bloc des déclarations
Règles de grammaire
Programme
A : B { /* action pour cette regle */ };

YACC
Exemple

- Exemple de spécification YACC *Génération d'une calculatrice*
- But: Intérprétateur de commande permettant de faire des opération telles que:
  2+5
  (1+2)
  *1*2*
  2(2+1)*
- Plus de simplicité:
  les nombres à plusieurs chiffres ne sont pas pris en charge

YACC I
Exemple

G(L) = <N,X,P,S> Avec

```
N : { ligne, commande, expr, terme, facteur }
X : { [0-9], \n, +, *, (, ) }
P : {
    ligne ->
        commande \n ligne
        | \n
    commande :
        expr
    expr :
        expr + terme
        | terme
    terme :
        terme * facteur
        | facteur
    facteur :
         ( expr )
        | [0-9]
}
```

YACC II
Exemple

```
S : ligne
```

YACC I
Exemple

Fonction yylex():

```
%{
    #include <ctype.h>
    #include <stdio.h>
    #include <stdlib.h>
%}

%token CHIFFRE

%%
ligne : commande '\n' ligne
    | '\n' { printf("Fin du programme\n"); exit(0); }
    ;

commande: expr { printf("Resultat: %d\n", $1); };


expr : expr '+' terme { $$ = $1 + $3; }
    | terme
```

YACC II
Exemple

```
    ;

terme : terme '*' facteur { $$ = $1 * $3; }
    | facteur
    ;

facteur : '(' expr ')' { $$ = $2; }
    | CHIFFRE
    ;

%%
int main(){
    yyparse();
}

int yyerror(char *s){
    printf("%s \n", s);
}


int yylex(){
```

YACC III
Exemple

```
    int c;
    c = getchar();
    if(isdigit(c)){
        yylval = c-'0';
        return CHIFFRE;
    }
    return c;
}
```

YACC
Spécifications

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

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

Plan

Antlr
Definition

- - Générateur de parseur public
- - LL(k)
- - Framework

Antlr
Quelques Fonctionnalités

- intègre la spécification entre une analyse lexicale et syntaxique.
- facilite la construction de l'arbre syntaxique.
- génère des parseurs de descente récursives en C et C++.
- facilite la gestion d'erreurs.

Antlr

Elements du langage:

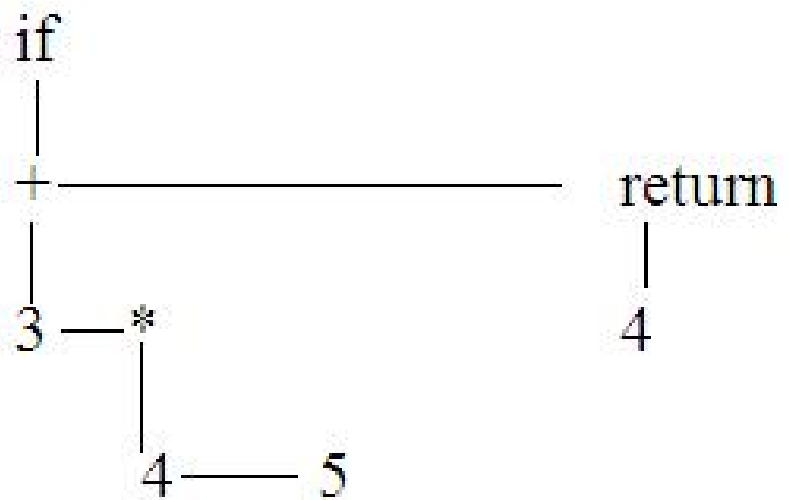
Elément de langages	Description	Exemple
Token	Commence par majuscule	ID
$\prec\prec \ldots \succ\succ$	Définie une action sémantique	$\prec\prec printf("\%S", a); \succ\succ$
(...)	Régle	$("int" \mid ID \mid storage\_class)$
(...)*	Closure	ID(" "ID)*
(...)+	Positive Closure	slist : $(stat \mid SEMICOLON)$
{...}	Optionnel	$\{ELSEstat\}$
$\prec\prec \ldots \succ\succ?$	Prédicat sémantique	type : $\prec\prec is\_Type(str) \succ\succ ?ID$
(...)?	Prédicat syntaxique	$((listEQ))?listEQlist \mid list$

Antlr I
Gestion de l'arbre

Voici un petit exemple de la constrcuction d'un arbre syntaxique :

```
if 3 + 4 * 5 then return 4;
```

L'arbre correspondant :



Antlr
Gestion d'erreurs

Deux mécanismes de gestion d'erreurs:

- Gestion automatique d'erreurs
- *Parser Exception Handling*

Antlr I

Exemple de grammmaire Antlr: Calculatrice simple

```
grammar GrammaireSimple;

options {
  language = Java;
}
tokens {
    PLUS     = '+' ;
    MINUS    = '-' ;
    MULT     = '*' ;
    DIV = '/' ;
}

@members {
```

Antlr II

Exemple de grammmaire Antlr: Calculatrice simple

```
    public static void Main(string[] args) {
        GrammaireSimpleLexer lex = new GrammaireSimpleLexer(new
        CommonTokenStream tokens = new CommonTokenStream(lex);

        GrammaireSimplecParser parser = new GrammaireSimplePars

        try {
            parser.expr();
        } catch (RecognitionException e)  {
            Console.Error.WriteLine(e.StackTrace);
        }
    }
}

 %PARSER RULES
```

Antlr III

Exemple de grammmaire Antlr: Calculatrice simple

```
expr     : term ( ( PLUS | MINUS )  term )* ;

term     : factor ( ( MULT | DIV ) factor )* ;

factor   : NUMBER ;


 %LEXER RULES

NUMBER   : (DIGIT)+ ;

WHITESPACE : ( '\t' | ' ' | '\r' | '\n'| '\u000C' )+    { $chan
fragment DIGIT   : '0'..'9' ;
```

Plan

Résultat

Yacc(LALR) vs Antlr(LL):

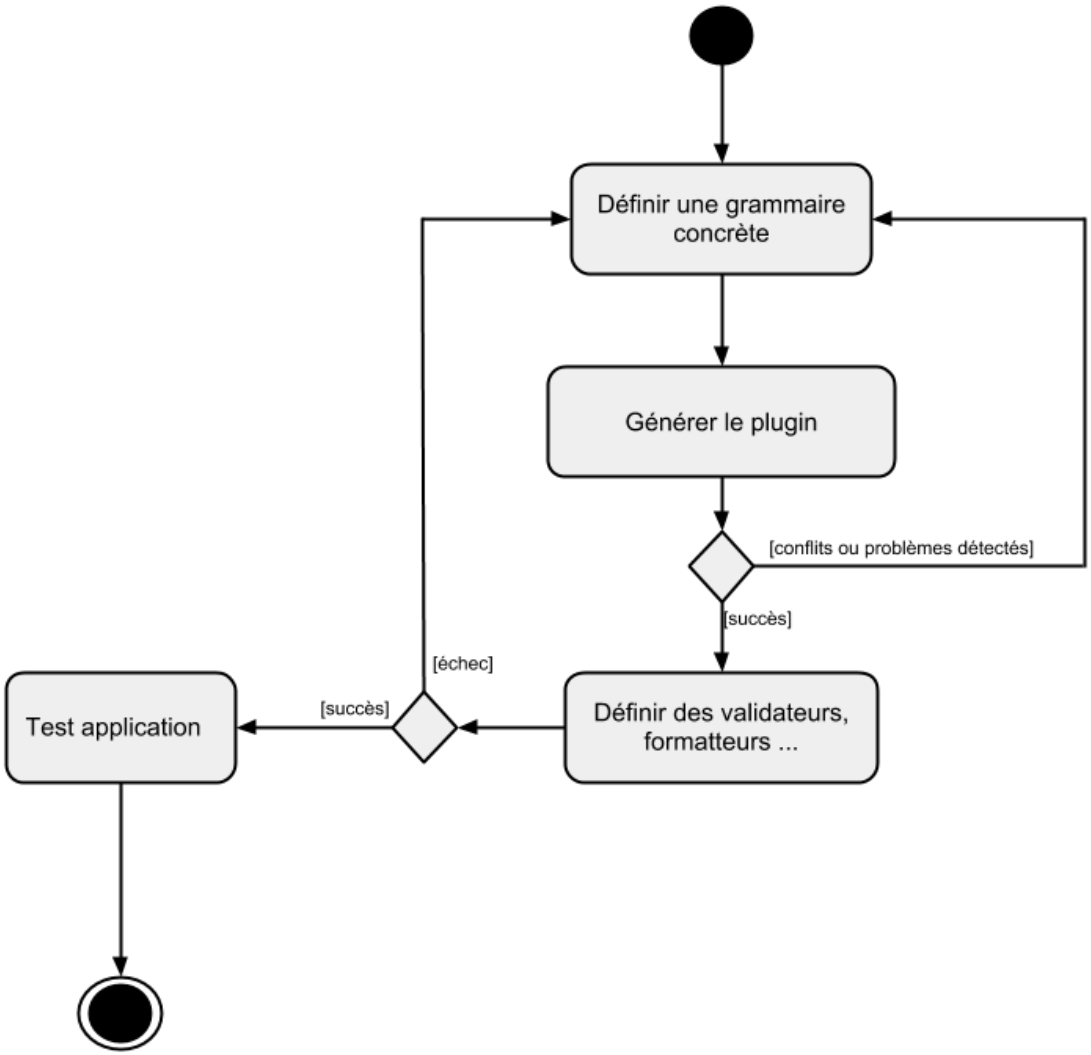
- Récursivité à gauche
- complexité d'utilisation
- Performance et flexibilité

Plan

Xtext
Définition

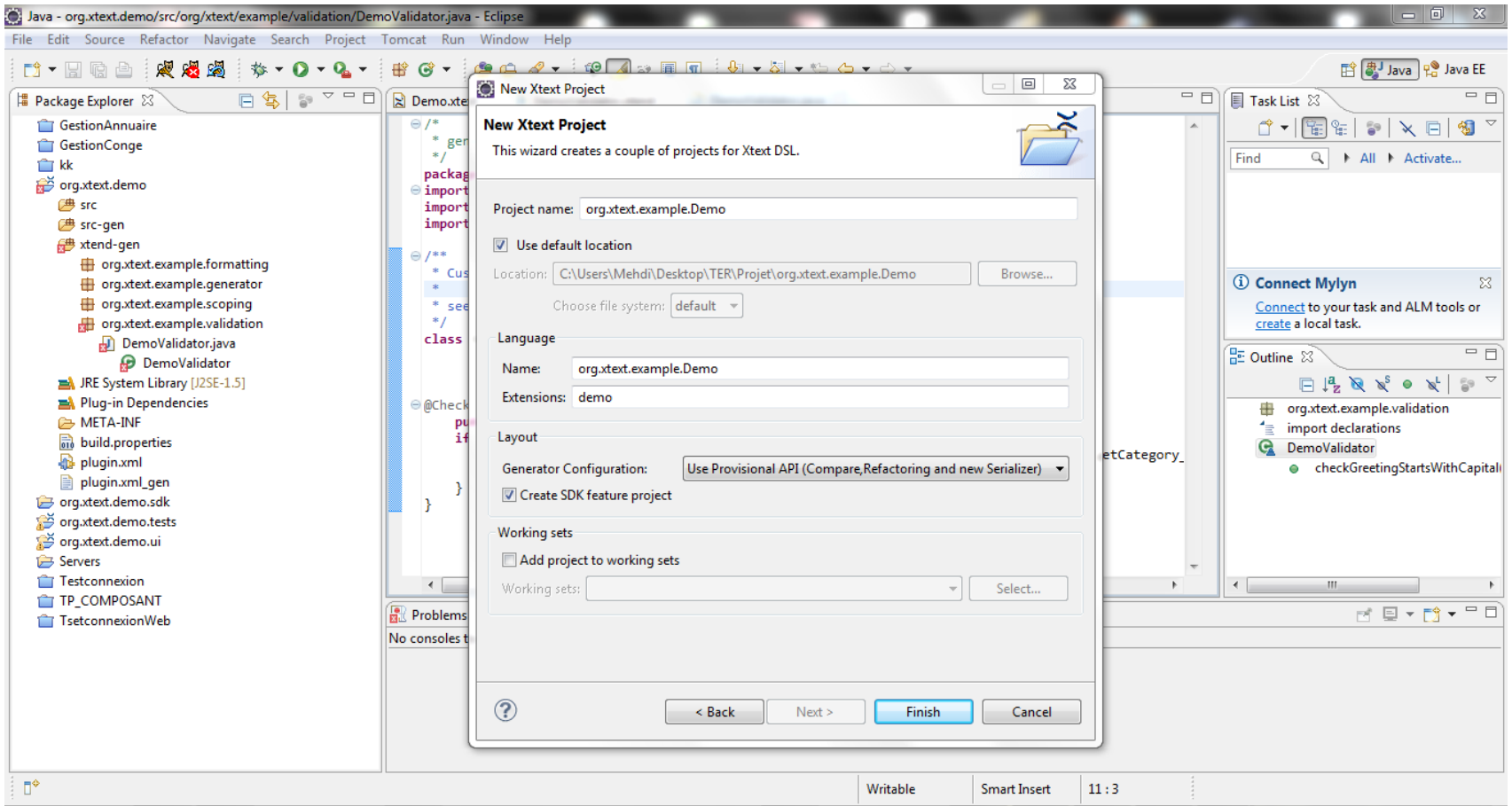
- - Framework Eclipse
- - Développement de langages de programmation et de DSL
- - Grammaire proche de celle de Antlr
- - Parser LL(*)

Xtext
Fonctionnement



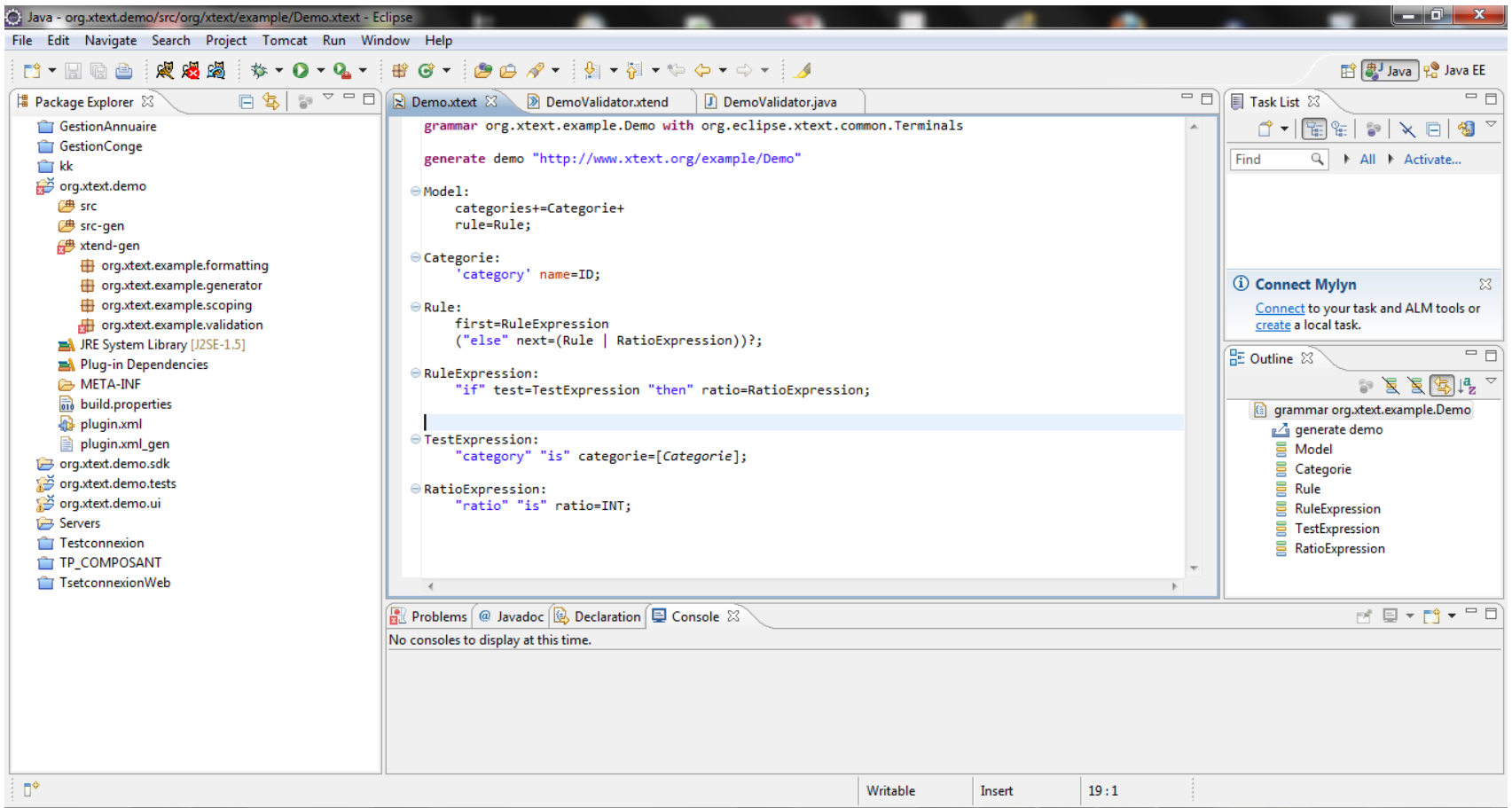
Xtext:Exemple

Etape 1: Création d'un projet Xtext



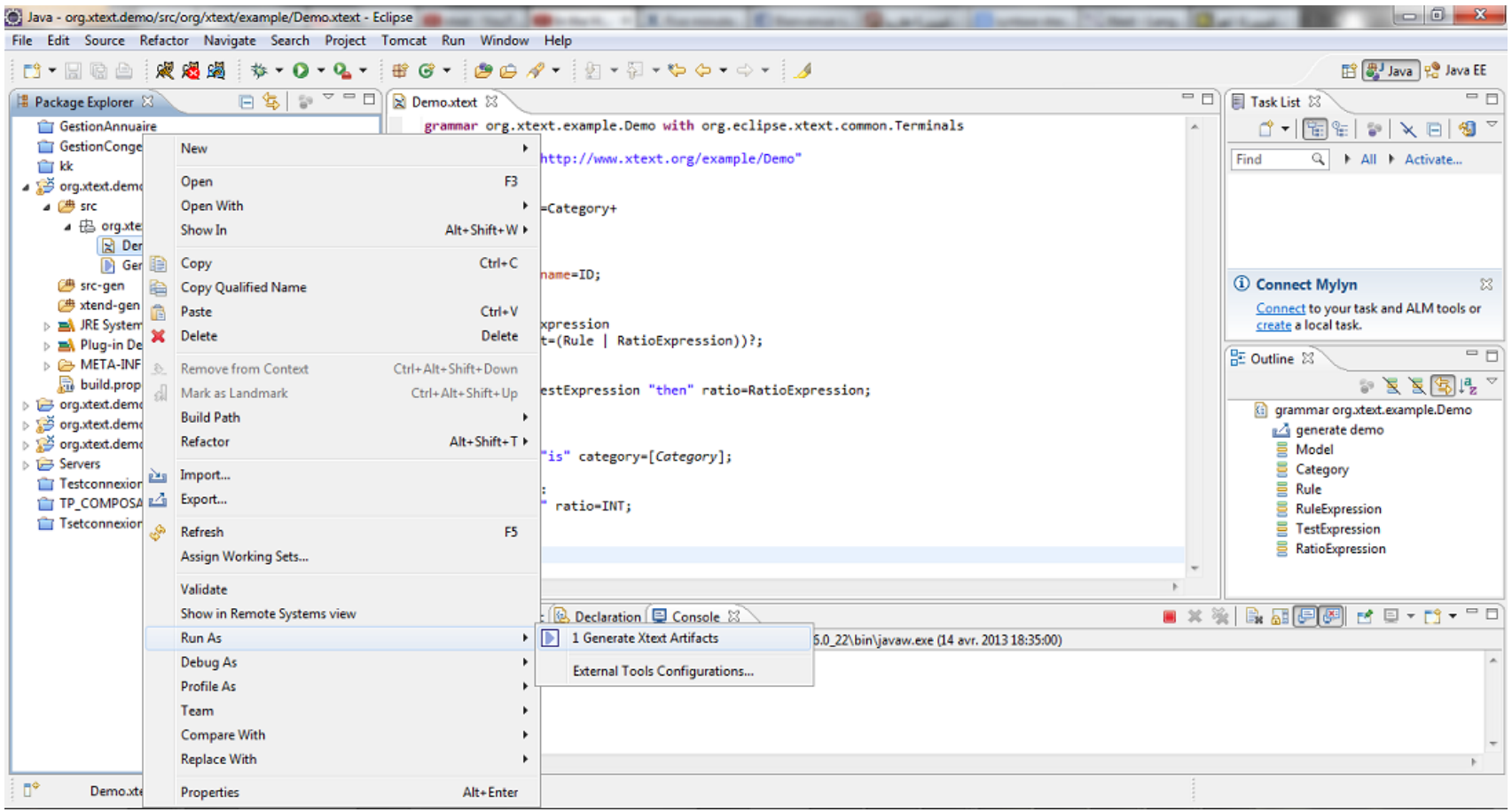
Xtext:Exemple

Etape 2: Definition du langage



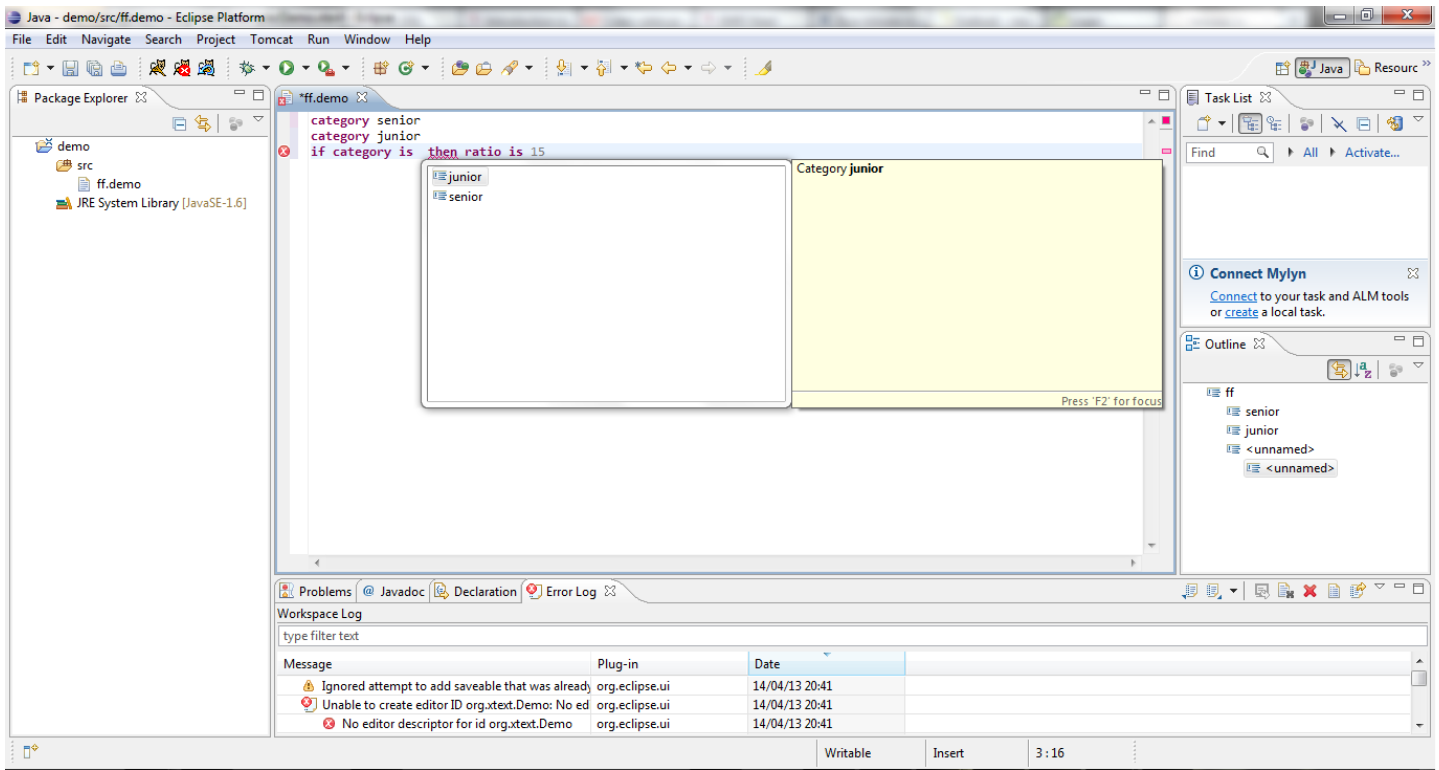
Xtext:Exemple

Etape 3: Generation des artifacts



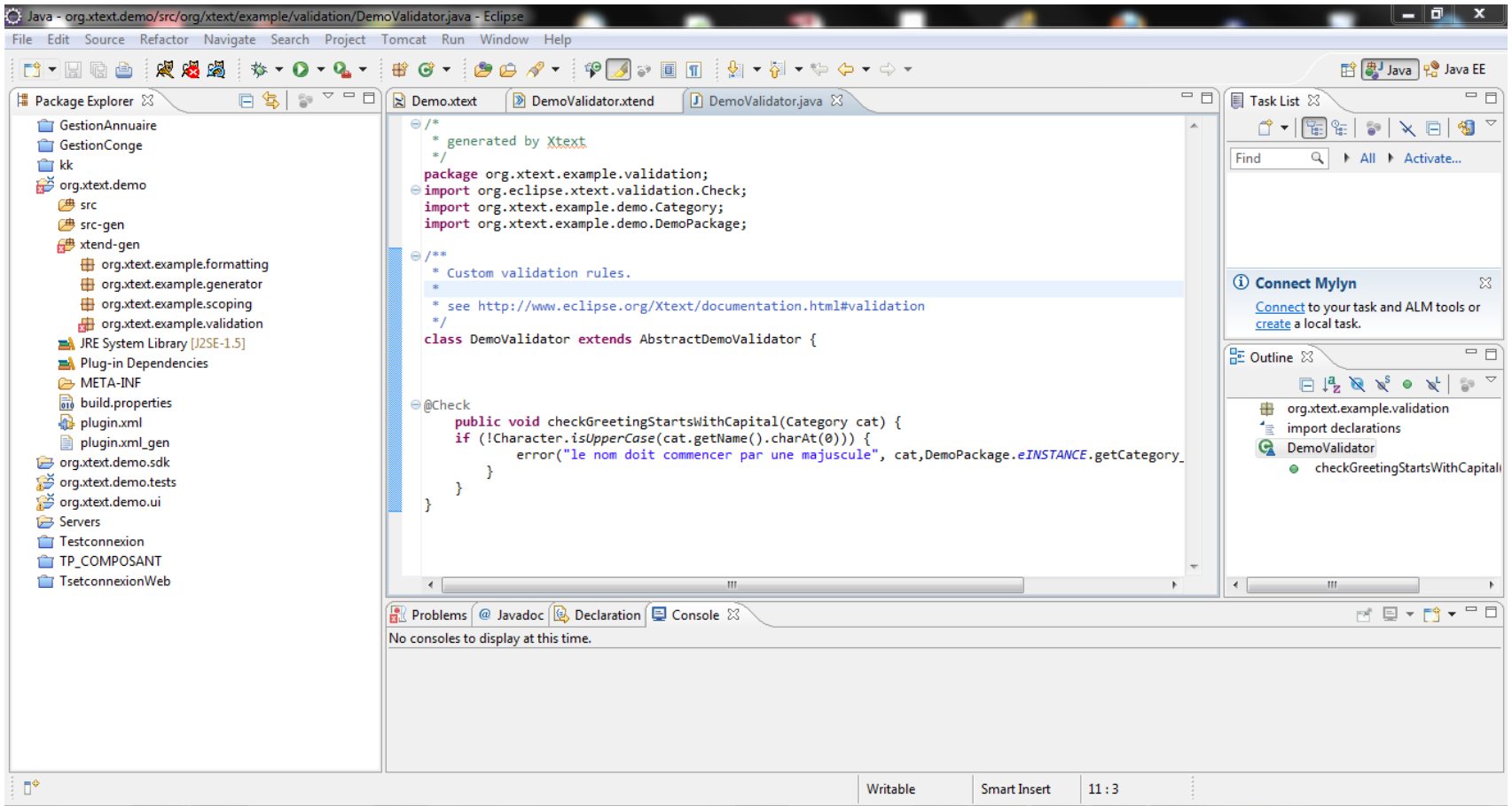
Xtext:Exemple

Etape 4: Faire un test



Xtext:Exemple

Etape 5: Ajouter une validation



Plan