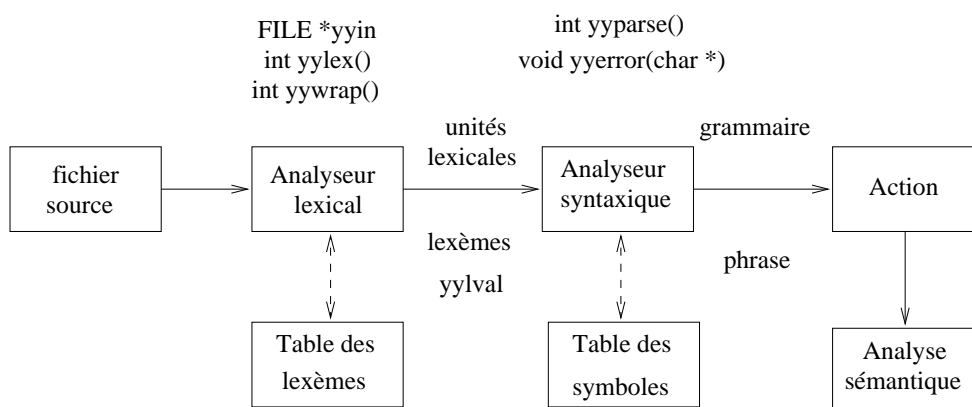


TP n° 2 - (3 heures) - *JFlex, byacc*

On rappelle que la phase d'analyse est décomposée comme suit :



On désire lire des fichiers contenant des clauses du calcul des prédicats et les représenter en mémoire. Les fichiers auront la forme suivante :

$p(X) \mid q(Y, f(Z))$.
 $\neg p(a)$.

On remarque que :

- les symboles de prédicat, de fonction et de constante sont en lettres minuscules
- les symboles de variable apparaissent en lettres majuscules
- le symbole \neg est représenté par -
- le symbole \vee est représenté par |
- chaque clause est terminée par un .

Exercice 1 - Table des lexèmes

Créer une classe `Lexem` qui contient une chaîne de caractères et dont la spécification est la suivante :

class Lexem
◇ <i>Attributs</i>
String string // chaîne de caractères
◇ <i>Méthodes</i>
Lexem(String s)
String getContent()

Créer ensuite une classe `LexemTable` qui permet de stocker les lexèmes de manière non redondante. On pourra par exemple dans un premier temps utiliser une liste :

class LexemTable	
◇ <i>Attributs</i>	
ArrayList	list // liste contenant les lexèmes
◇ <i>Méthodes</i>	
	LexemTable()
Lexem	add(String s)

La méthode `add` ajoute un lexème dans la table si celui-ci n'est pas déjà présent, sinon on retourne le lexème existant.

Exercice 2 - Table des Symboles

Créer une classe `Symbol` qui permettra de représenter un symbole (Variable, Constante, prédicat ou fonction). La spécification est la suivante :

class Symbol	
◇ <i>Attributs</i>	
Lexem	lexem // lexème associé au symbole
int	type // type de symbole (VARIABLE, CONSTANT, COMPLEX)
int	arity // arité pour les symboles COMPLEX
int	identifiant // identificateur de symbole
◇ <i>Méthodes</i>	
	Symbol(Lexem l)
Lexem	getLexem()
int	getType()
int	getArity()
int	getIdentifiant()
String	getName()
void	setIdentifiant(int id)
void	setType(int t)
void	setArity(int a)
boolean	isVariable()
boolean	isConstant()
boolean	isComplex()

Créer ensuite une table des symboles qui permet de stocker les symboles. On utilisera une structure de table de hachage pour stocker et accéder facilement aux symboles (`HashMap`). Comme pour la table des lexèmes, le symbole est unique.

class SymbolTable	
◇ <i>Attributs</i>	
HashMap	map
◇ <i>Méthodes</i>	
	SymbolTable()
Symbol	add(Lexem l, int type)

Exercice 3 - Analyseur lexical

Ecrire la définition de l'analyseur lexical qui sera généré par JFlex et appeler le fichier correspondant : `scanner.lex`.

Exercice 4 - Analyseur Syntaxique

Ecrire la grammaire reconnue par byacc et qui permet de lire un fichier contenant des clauses du calcul des prédicats (fichier `parser.y`).

Exercice 5 - Téléchargement et Installation de JFlex/byacc

- télécharger JFlex et byacc (sur mon site)
- installer les logiciels
- créer un fichier `ant` pour lancer la compilation du *scanner* et du *parser*