NF11 – TP2 : GENERATION D'ANALYSEUR LEXICAL ET SYNTAXIQUE (4 SEANCES)

GENERALITES

OBJECTIFS

Créer une grammaire du langage Logo et une représentation intermédiaire d'un programme Logo.

Générer un analyseur syntaxique de cette grammaire.

Créer un interpréteur graphique du langage Logo.

Crée une table des symboles d'un programme.

Vérifier quelques conditions sémantiques.

OUTILS

Editeur de grammaires, générateur d'analyseur lexical et syntaxique (lexer – parser - parser d'arbre) à partir de grammaires AntLR Version 3.3.

Outil : AntlrWorks version 1.4.2, installé sur les stations
Option : téléchargeable depuis tp/Espace projets –TP/ Outils TP2/antlrworks1.4.2.zip du site moodle NF11 ; exécuter le fichier antlrworks.bat

Plug-in Eclipse : ANTLR IDE, installé sur les stations
Pour transformer un projet java de eclipse en projet ANTLR :
Configure/convert to ANTLR project

Matériel initial pour le TP2
Site moodle NF11, Espace projets –TP/Outils TP2/tp2-2011.zip
Ce fichier contient l'embryon des grammaires LOGO et la librairie antLR

DOCUMENTATION ANTLR

Site :	http://www.antlr.org/
Documentation :	ANTLR v3 pour commencer, site ANTLR, ANTLR v3 documentation,
API :	ANTLR v3 API documentation, site ANTLR
Livre :	The definitive ANTLR reference, Bibliothèque UTC

DOCUMENTATION LOGO

Manuel Logo : Espace projets –TP/Documentations/Manuel LOGO, site moodle NF11

DOCUMENTATION JAVA

http://java.sun.com/docs/books/tutorial/index.html

ETAPE 1: INTRODUCTION

CONFIGURATION D'ECLIPSE

- Créer un projet Java dans l'environnement Eclipse (ex : tp2)
- Télécharger le fichier tp2-2011.zip à partir du site NF11 ;
- Décompacter le répertoire src de l'archive dans le répertoire source (tp2/src) d'Eclipse ;
- Créer un répertoire lib sous ce projet (tp2/lib) ; décompacter le répertoire lib de l'archive dans le répertoire lib du projet Eclipse ;
- Installer la librairie ANTLR (antlr-3.3-complete.jar) dans le classpath du projet
    - Build Path/add to Build Path
- L'ensemble des fichiers doit ressembler à la Figure 1.

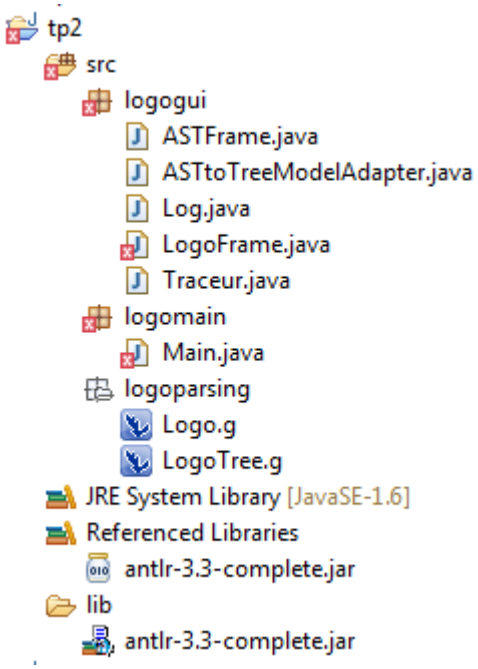


FIGURE 1 : APERÇU DU PROJET

- Transformer le projet java en projet ANTLR : Configure/convert to ANTLR project

ETUDE DE LA GRAMMAIRE INTRODUCTIVE

- Ouvrir la grammaire Logo.g du répertoire logoparsing.
- Ajouter le token TG au lexer et l'instruction TG INT au parser dans la grammaire Logo.g.
- Sauvegarder. Les classes lexer et parser doivent être générées dans le même répertoire que la grammaire.
- Décommenter la méthode parse1() dans le main de la classe logomain.Main et exécuter la. Entrer un programme tel que : AV 100 TG 90 AV 100.
- Décommenter la méthode parse2() dans le main de la classe logomain.Main et exécuter la. Entrer un programme tel que : AV 100 TG 90 AV 100.
- Ouvrir la grammaire LogoTree.g.
- Ajouter le parsing d'un arbre de la forme : ^(TG INT) ; faire générer le parser d'arbre ; ajouter une méthode tg à la classe logogui.Traceur.
- Décommenter et exécuter la méthode ui de la classe logomain.Main.
- Ecrire un programme logo simple dans le panel de gauche et tester. Cette application permet de visualiser le résultat de l'interprétation du programme Logo.
- On peut également travailler sur les fichiers grammaires à partir de l'éditeur AntlrWorks.

ETAPE 2 : LEXER, PARSER, INTERPRETEUR DU LANGAGE LOGO

Etudier le document Manuel Logo pour avoir une idée de ce langage.

De façon cyclique :

> ajouter une règle au lexer, au parser et au parser d'arbre du langage LOGO afin d'ajouter une fonctionnalité.
>
> Faire générer les classes ;
>
> TESTER chaque ajout un à un.

Il faudra également compléter la classe logogui.Traceur et dans les étapes 3 et 4 créer éventuellement d'autres classes.

Il ne s'agit pas de représenter intégralement la grammaire de Logo, mais d'implémenter les éléments les plus caractéristiques.

CONSIGNES

Ne pas générer les synonymes des commandes logo. La forme courte suffit.
Ne générer que quelques fonctions mathématiques. Deux ou trois suffisent.
Utiliser l'écriture arithmétique habituelle (m + n) et non préfixée (somme m n).

REPRESENTER :

- Les principales commandes de la tortue (version courte).
- Les expressions arithmétiques (des exemples se trouvent sur le site AntLR).
- les expressions booléennes.
- les identificateurs, l'affectation.

EN COMPLEMENT

- Construire la table des variables du programme principal dans le parseur et la communiquer au parseur d'arbre. Il faudra modifier la méthode runParser de la classe logogui.LogoFrame.
- Indiquer une erreur par un avertissement dans le panel de Log et éventuellement par un arrêt de l'interprétation au niveau du parser. On utilisera l'attribut booléen valide déclaré dans le parseur (voir la partie @members de la grammaire).
- Adapter le calcul d'expressions arithmétiques en ajoutant la possibilité d'utiliser la valeur d'une variable.

VALIDITE DU PARSING

La classe parser possède un champ booléen appelé `valide` destiné à déterminer la validité du parsing. Par défaut à `true`, il peut être mis à `false` pour ne pas générer une analyse à partir de la structure d'arbre intermédiaire.

Le panneau de commande possède un panneau de Log. Il est géré par la classe logogui.Log. Il est possible d'écrire dans ce panneau en utilisant les méthodes append et appendl. Il suffit d'insérer dans la grammaire une action du type : `{Log.appendnl("Programme principal");}`

Signaler :

- Utilisation d'une variable non initialisée

ETAPE 3 : STRUCTURES DE CONTROLE

REPRESENTER

- L'alternative (si).
- La répétition (répète, tant que).
- Les variables locales à un bloc : leur déclaration utilise le mot-clé LOCALE. Dans le cas contraire une variable est considérée globale au programme.

ETAPE 4 : PROCEDURES ET FONCTIONS

REPRESENTER

- Les procédures.
- Les fonctions.
- Les procédures récursives

COMPLETER

- La table des symboles.

VERIFIER

- L'arité d'une procédure ou d'une fonction.
- Qu'une variable locale ne peut avoir le même nom qu'un paramètre.
- Qu'une variable globale dans une procédure ou une fonction.

ORGANISATION DU TRAVAIL

L'étape 1 doit durer une heure.

L'étape 2 doit durer 2 semaines (1 séance encadrée + travail personnel).

L'étape 3 doit durer 2 semaines (1 séance encadrée + travail personnel).

L'étape 4 doit durer 4 semaines (2 séances encadrées + travail personnel).

EVALUATION

SOUTENANCE ORALE (DERNIERE SEANCE)

Faire une démonstration de l'interpréteur à partir d'un ou plusieurs programmes LOGO. Illustrer les différentes caractéristiques implémentées dans les grammaires.

DOCUMENTS A FOURNIR APRES L'ETAPE 2

Fournir les fichiers contenant les grammaires : lexer, parser, et parser d'arbre.

DOCUMENTS A FOURNIR EN MEME TEMPS QUE LA SOUTENANCE ORALE

Fournir les fichiers contenant les grammaires définitives : lexer, parser, et parser d'arbre.

Faire un compte-rendu de quelques pages explicitant :

- la modélisation de la table des symboles et le fonctionnement de l'interpréteur (ex : structure des tables, appel et interprétation des procédures, passage de paramètres, etc.)
- la façon dont les procédure ont été traitées (passage des paramètres, appel d'une procédure dans une procédure, procédure récursive, fonction).
- les éléments non traités

COMPLEMENT : REPRESENTATION GRAPHIQUE

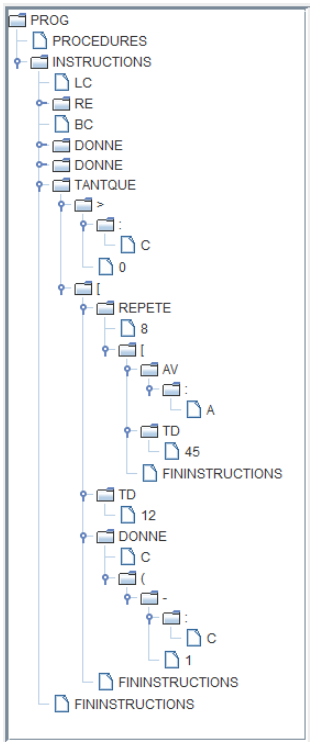
Pour réaliser un tracé graphique dans un panel, il est nécessaire d'obtenir un objet de type java.awt.Graphics à partir du panneau lui-même. C'est l'objectif de la méthode initialize ajoutée à la classe du parseur d'arbre. Les constructions graphiques sont confiées à un objet de type Traceur auquel est transmis le Graphics. A chaque fois qu'une application décide de réaliser un tracé dans un panneau, elle récupère l'objet Graphics de celui-ci et appelle la méthode initialize du parseur d'arbre.

EXEMPLE DE JEU D'ESSAI

PROGRAMME SOURCE

```
LC RE 100 BC
DONNE "A 70
DONNE "C 30
TANTQUE :C >0 [
   REPETE 8 [AV :A TD 45]
   TD 12
   DONNE "C (:C - 1)
 ]
```

ARBRE ABSTRAIT CONSTRUIT PAR LA GRAMMAIRE LOGO ET VUE



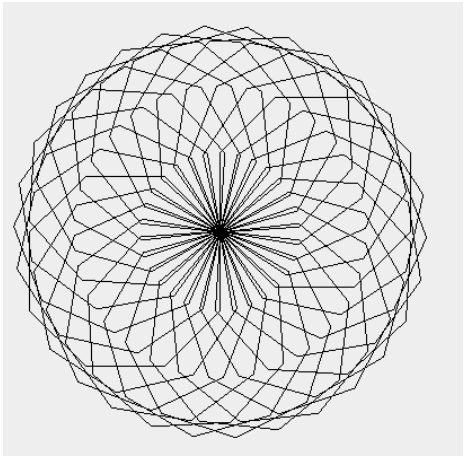


FIGURE 2 : ARBRE AST ET VUE