Projet de Compilation
Langage Tiger

Rapport de projet 2

Auteurs
Thibaut CAGNON
Salim EL OUAFKI
Baptiste GHEYSENS

Responsable de module
Suzanne COLLIN

Octobre 2022-Janvier 2023

1 Introduction

L'objectif du projet de compilation est, pour les étudiants des filières IL, LE et ISS, d'écrire le compilateur d'un langage dit de "haut niveau", comme le language "Tiger". Pour les élèves des filières IAMD et SIE, l'objectif du projet est d'écrire la grammaire du langage afin d'en faire une analyse syntaxique et sémantique. Ce rapport d'activité rend compte, dans un premier temps, de la gestion du projet, et dans un second temps, de la conception du projet, des diffcultés rencontrées et des futures étapes du projet.

2 Conception

2.1 Résumé

L'écriture de la grammaire étant complétée, nous avons réalisé l'implémentation de l'arbre syntaxique abstrait et sa visualisation. Il reste quelques détails à gérer, notamment le nom de certains noueds afin qu'ils soient plus explicites. Nous allons ensuite passé à l'implémentation de la table des symboles et des contrôles sémantiques.

2.2 Construction de l'Arbre Syntaxique Abstrait

Nous avons construit un Arbre Syntaxique Abstrait (AST) pour simplifier l'analyse sémantique et rendre l'arbre plus simple à comprendre.

Pour cela, nous avons construit une interface AST qui représente la structure arborescente de l'AST et une interface ASTVisitor dont les méthodes permettent de parcourir des structures arborescentes. Cela permet de différencier la classe qui parcourt la structure arborescente de la structure elle-même. L'interface décrit une méthode visit pour chaque classe de la structure arborescente.

Les classes de la structure arborescente implémentent l'interface AST. De plus, les classes visitées implémentent une méthode accept qui prend un visiteur en argument et qui permet au compilateur de décider du type du noeud en entrée et d'appeler la bonne méthode visit.

Afin de pouvoir visualiser l'arbre syntaxique abstrait, nous avons créer une classe GraphVizVisitor implémentant l'interface ASTVisitor. Elle permet de créer les noeuds et les transitions de l'arbre pour chaque classe de la structure arborescente.

Des exemples de code source testant chaque règles ont été ajoutés afin de vérifier l'implémentation de l'AST.

```
(let var N := 8
in
end;
let var M := 8
in
if "bonsoir" then print("M is zero") else 0;
let
var N := 8
var N := 5
in
end
end)
```

Figure 1: Exemple de programme

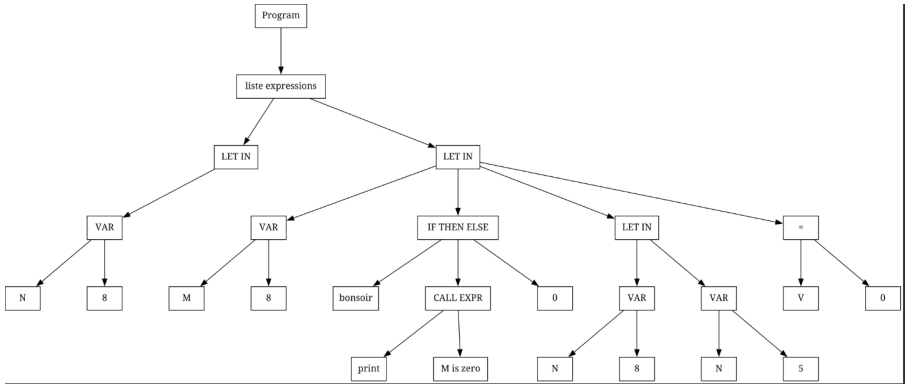

Figure 2: Exemple d'AST

3 Difficultés rencontrés

Réaliser l'AST n'a pas posé de difficultés importantes, si ce n'est une certaine réflexion sur la gestion des $+$ et $*$ dans la grammaire.

La seule difficulté a été de trouver le temps de le terminer avant la date limite puisque celle-ci arrivait pendant la période des partiels, alors que d'autres projets étaient également en cours.

4 Futures étapes du projet

4.1 Table des symboles

Les prochaines étapes du projet consistent à réaliser l'implémentation de la TDS.

4.2 Contrôles sémantiques

Nous devons effectuer l'analyse sémantiques du langage CIRC-C. L'analyse sémantique consiste à effectuer des vérifications sur la sémantique du langage et à construire la table des symboles. Notre

objectif est d'écrire 5 à 6 contrôles sémantiques par membres. Nous allons rédiger une liste de contrôles sémantiques à implémenter et les répartir entre nous.

Nous avons réfléchi à plusieurs contrôles sémantiques, comme la vérification des types, l'affectation d'une variable non initialisée et la portée des déclarations.