PCL
Projet Compilation des Langages
2023-2024

17 Janvier 2024

AULAGNIER Mathis
BOUCHADEL Maxence
CABILLOT Matthias
VATRY Etienne

Table des matières

1 Introduction

Dans le contexte du module PCL, nous avons été chargés de réaliser un compilateur pour le langage canAda. Ce rapport aborde la première partie de ce projet, qui se concentre spécifiquement sur le développement d'un analyseur lexical, d'un analyseur syntaxique et la création d'un arbre AST (Abstract Syntax Tree).

Afin de réaliser ce compilateur, nous avions la liberté de choisir notre langage de programmation. Nous avons opté pour Python comme langage, principalement pour sa facilité d'utilisation et ses bibliothèques (Re pour l'utilisation d'expression régulière par exemple). Cette décision s'est basée sur notre maîtrise collective de Python, nous permettant de démarrer efficacement ce projet complexe.

Notre compilateur doit fonctionner comme ceci : l'analyseur lexical est destiné à identifier les tokens du langage canAda. Ensuite, l'analyseur syntaxique prend le relais pour s'assurer que les programmes respectent la grammaire du langage (grammaire que nous avons optimisé). Parallèlement, la construction de l'arbre AST nous aide à représenter la structure du programme de manière visuelle et hiérarchique, facilitant les étapes de compilation ultérieures.

Ce rapport détaille les méthodes que nous avons employées, les obstacles rencontrés et les stratégies mises en place pour les surmonter. Notre but est de fournir un aperçu clair de notre démarche et de montrer comment nous avons appliqué nos compétences en informatique pour réaliser ce projet.

2 Gestion de projet

2.1 Organisation générale

Afin de réaliser ce projet, nous avons essayé d'appliquer au mieux les différents éléments vu en cours de gestion de projet en première année. Pour commencer notre groupe était composé de quatre membres :

— Mathis AULAGNIER
— Maxence BOUCHADEL
— Matthias CABILLOT
— Etienne VATRY

Nous avons au lancement du projet désigné Mathis AULAGNIER en tant que chef du projet. Il avait pour rôle de motiver les membres lors de la réalisation du projet en leur donnant les différentes tâches à réaliser. Il avait de plus la mission de fixer et préparer les réunions en avance pour qu'elles soient les plus productives possible et d'assurer leur bon déroulement en les dirigeants. Le chef de projet s'occupait aussi de réaliser les comptes rendus et la todo list pour la prochaine réunion afin de garder une trace.

Afin d'organiser efficacement notre travail, nous avons établi une liste des tâches à chaque réunion qui était ensuite attribué à chaque personne, basée sur les échéances fixées . Par exemple, pour la semaine 50, il nous était demander de terminer l'analyseur lexical, tandis que l'analyseur syntaxique devait être presque fonctionnel.

Au début, nous avons tous contribué individuellement à l'élaboration de l'analyseur lexical, afin de nous familiariser avec le projet. Une fois cette phase d'initiation terminée, nous avons combiné nos efforts pour finaliser cet analyseur. Pour la seconde partie, concernant l'analyse syntaxique, nous nous sommes divisés en deux groupes : Maxence et Mathis étaient chargés d'optimiser la grammaire pour la rendre la plus compatible possible avec la forme LL(1), en s'appuyant sur les méthodes étudiées en cours de Traduction et sur les enseignements du Dragon Book. Parallèlement, Matthias et Etienne se sont concentrés sur la conception du code et sur la réflexion autour de la gestion et de la création de l'arbre syntaxique.

Ainsi grâce à cette organisation chacun savait exactement ce qu'il avait à faire et ce qu'il devait modifier.

2.2 Les moyens de communication

Afin de pouvoir assurer une bonne communication dans le groupe, différents moyens ont été mis en place. Nous avons commencé par la création d'un dépot github avec une arborescence hiérarchisé permettant la création, l'échange et modifications simples des différents documents nécessaires au projet autre que le code. Ceci a donc permis la création de la todo list que chacun pouvait consulter et modifier simplement, mais également la consultation des comptes rendus ainsi que des préparations des réunions contenant l'ordre du jour, auxquels chacun pouvait rajouter des éléments qu'il souhaitait évoquer en réunion.

Nous avons également créé un groupe Discord, permettant de communiquer à la fois par écrit et par vocal. Ce moyen nous a été très utile pour s'échanger des informations et pour pouvoir travailler ensemble lorsque nous le désirions .

Pour ce projet, le fait d'appartenir tous au même groupe de TD a grandement facilité notre communication. Cette situation a contribué à prévenir les problèmes de coordination (éviter les "effets tunnel"), et a simplifié l'organisation de nos réunions, en nous permettant d'aligner plus aisément nos emplois du temps respectifs.

2.3 Organisation des réunions

Dans le cadre de notre projet, nous avons accordé une importance particulière à l'organisation de nos réunions. Notre objectif était de faire un point de manière hebdomadaire, soit virtuellement sur Discord soit physiquement à l'école, profitant souvent des pauses pour se retrouver.

Ces rencontres régulières nous permettaient de faire le point sur l'avancement du projet, d'aborder les difficultés rencontrées et de discuter des étapes suivantes. De plus, à chaque fois qu'une avancée significative était réalisée, nous planifions une réunion plus longue afin d'en parler entre nous de manière posé et claire.

Ces sessions étaient également l'occasion de mettre à jour notre liste de tâches, assurant ainsi une progression continue et coordonnée du projet. Cette approche structurée nous a permis de maintenir un rythme de travail efficace et de rester constamment alignés sur nos objectifs communs.

3 Le compilateur

3.1 L'analyse Lexicale

L'analyseur lexical assume une position cruciale en tant que première étape incontournable du processus de compilation pour le langage canAda. Son rôle central consiste à lire le fichier source caractère par caractère, identifier les tokens qui composent le langage, et les transmettre à l'analyseur syntaxique. Implémenté sous la forme d'un automate, l'analyseur lexical a été spécifiquement adapté pour répondre aux exigences du projet.

3.2 Fonctionnement de l'Analyseur Lexical

L'automate de l'analyseur lexical s'acquitte de diverses tâches essentielles, débutant par l'élimination des commentaires. Les commentaires détectés sont éliminés, simplifiant ainsi le traitement du code source. Ensuite, l'analyseur lexical procède à la suppression des espaces, tabulations et retours à la ligne, normalisant ainsi la présentation du code et facilitant le processus d'analyse.

Éventuellement, l'analyseur lexical prend en charge l'affichage des erreurs lexicales en signalant des anomalies potentielles telles que la présence de caractères inconnus, d'identificateurs trop longs ou de constantes mal formées. Des limites ont été fixées de manière arbitraire, comme une taille maximale de 40 caractères pour les identificateurs et de 10 caractères pour les constantes.

```
#Definition de la taille max d'un identificateur
MAX_IDENT_SIZE = 40
#Definition de la taille max d'une constante
MAX_CONST_SIZE = 10
```

Par la suite, l'analyseur lexical découpe le code source en "tokens". Ces unités lexicales identifiées sont codées de manière spécifique pour faciliter la manipulation ultérieure. Cette notation attribue des codes numériques aux opérateurs, mots-clés, identificateurs et constantes. L'objectif sous-jacent à cette démarche est l'optimisation de l'efficacité du compilateur. L'utilisation de codes numériques pour représenter les unités lexicales offre une amélioration significative de l'efficacité du traitement. Cette approche permet une comparaison rapide d'entiers plutôt que des chaînes de caractères, contribuant ainsi à l'amélioration globale des performances du compilateur.

En conclusion, l'analyseur lexical joue un rôle essentiel en préparant le

terrain pour l'analyse syntaxique. Son adaptation précise aux spécificités du langage canAda garantit un fonctionnement robuste et efficient, tout en offrant une base solide pour les étapes ultérieures du processus de compilation.

3.3 L'analyse Syntaxique

3.4 Les arbres

3.4.1 Problématique

3.5 Présentation générale

3.6 Difficultés rencontrés

3.7 Points d'améliorations

4 Conclusion

5 Annexes

5.1 Exemple d'utilisation du programme