Rapport de projet d'analyseur syntaxique de Little ADA

Joann Vetter

5/01/2022

Nous avons choisi pour ce projet d'utiliser le langage C pour avoir une vision différente de celle vue en cours et pour parfaire nos connaissances en C. Même si le langage n'est pas optimal pour réaliser un analyseur syntaxique, il se prête bien au jeu de la reconnaissance d'expressions régulières grâce à l'implémentation GNU de lex et yacc, flex et bison.

Il fallait pouvoir générer un parser de fichier Little Ada pour dans un premier temps relever les erreurs de syntaxes et pouvoir construire l'arbre de syntaxe abstraite du fichier lu. Pour ce faire, on a utilisé lex et bison qui permettent de générer ce parser à l'aide d'un fichier .l qui va scanner les mots pour vérifier que ceux-ci appartiennent bien au langage qu'on veut lire. Il a donc fallu d'abord écrire le .l, un travail de recherche d'expressions régulières principalement notamment pour reconnaitre les différentes constantes ainsi que les identifiants.

Une fois les expressions régulières trouvées, il fallait expliciter la grammaire dans le fichier .y. Pour ce faire, les règles de grammaire ont été écrites dans ce fichier qui est particulièrement long en raison du nombre de cas possibles pour chaque ligne qui est relativement élevé. Il a fallu factoriser le code autant que possible pour garder un ensemble à peu près lisible, puis résoudre les différents conflits, ce qui a pris du temps.

Ensuite, après vérification des différents tests (certains tests KO marchent car nous n'avons pas eu le temps de finir le projet et de réaliser la fonction qui s'assure de la bonne utilisation des variables par exemple, etc), il a fallu trouver une structure pour représenter l'arbre. Pour ce faire, une structure assez lourde a été mise en place dans laquelle chaque noeud est représenté par une structure qui va chercher d'autres structures pour détailles sa composition.

Cette solution est beaucoup moins naturelle en C qu'en Ocaml ce qui nous a fait perdre beaucoup de temps. Une fois les structures déclarées, il faut remplir le fichier y avec les bonnes instructions pour pouvoir remplir l'arbre en fonction de ce qu'il y a dans les fichiers que l'on analyse, et les fonctions associées auraient étés réalisées dans le fichier structures.c.

On notera que les tests KO qui passent et ne devraient pas sont les suivants :

- AffConst.ada et AffIn.ada
- Toutes les erreurs de scope
- UnknownVariable.ada
- WrongCte4 et WrongCte5
- WrongId1 et WrongId3
- WrongOp1 et WrongOp2

Il est évident que si nous étions allés jusqu'à la question 6 ces tests la ne passeraient pas non plus et le parser serait complet, il arrive toutefois à détecter beaucoup d'erreurs et est complet syntaxiquement et sémantiquement.

Le projet n'est pas fini car comprendre comment implémenter un parser en C a pris beaucoup de temps, néanmoins le parser est fonctionnel et permet d'analyser syntaxiquement et sémantiquement un fichier little ADA pour savoir s'il est correct ou non. Les tests ont étés automatisés grâce à un petit script.