Sorbonne Université

Faculté des Sciences et Ingénierie

Parcours Science et Technologie du Logiciel
(STL)

Rapport de projet : Analyse des Programmes et Sémantique (APS)

Auteurs

Abdelkader Boumessaoud
Zaky Abdellaoui

Encadrant

Romain Demangeon
Loĺc Sylvestre
Hector Suzanne

Table des matières

1 Le protocole a suivre pour utiliser le projet

1.1 Outils nécessaires

— Pour la compilation OCaml : ocamllex et ocamlyacc.
— Pour le typeur : SWI-Prolog

1.2 Compilation

On passe par le Makefile pour compiler le programme écrit. Pour l'exécuter, il suffit d'ouvrir un terminal et de naviguer jusqu'au dossier contenant le fichier Makefile. Ensuite, il suffit de taper la commande 'make' pour lancer la compilation.

1.3 Exécution

Chaque version d'APS est composée de trois modules qui doivent être testés séparément. Voici les commandes pour tester chaque module :

Module d'analyse syntaxique et de traduction en term prolog : ./prologTerm path/progXXX.aps
Typeur (écrit en prolog) exécutable avec swipl : ./typrog path/progXXX.aps
Module fournissant l'évaluateur : ./evaluateur path/progXXX.aps

Il est important de remplacer "progXXX" par le nom du fichier APS que vous voulez tester. Ces commandes permettent de vérifier le bon fonctionnement de chaque module et de s'assurer que le code APS est valide.

2 La portee globale du projet

Nous avons implémenté toutes les fonctionnalités d'APS0 jusqu'à APS3 et nous avons réussi à passer quelques tests pour seulement 3 versions (APS0, APS1 et APS1a). Nous avons également développé un évaluateur, un analyseur lexical (lexer), un analyseur syntaxique (parser) et un générateur de termes Prolog en OCaml. La partie typage est également terminée, et nous avons utilisé Prolog pour cela.

APS0 : fonctionnalités implémentées et testées.
APS1 : fonctionnalités implémentées et testées.
APS1a : fonctionnalités implémentées et testées.
APS2 : fonctionnalités implémentées mais non testées/debugées.
APS3 : fonctionnalités implémentées mais non testées/debugées.

3 Choix dimplementations

Notre choix s'est porté sur le langage Ocaml. Ce langage de programmation est multi-paradigme, ce qui signifie qu'il peut être utilisé pour implémenter des programmes dans différents styles tels que la programmation fonctionnelle, impérative ou orientée objet.

De plus, Ocaml possède une fonctionnalité appelée pattern matching, qui est très puissante et permet de simplifier considérablement le code. Cette fonctionnalité permet de rechercher des motifs dans les données, et de les utiliser pour exécuter différentes actions en fonction des cas rencontrés.

En choisissant Ocaml, nous sommes convaincus que nous pourrons développer un code efficace, facile à comprendre et à maintenir