UNIVERSITÉ DE RENNES 1

Calculabilité

Rapport de travaux pratiques sur les interpréteurs

Glen Le Baill

Antoine Bouffard

Introduction

L'objectif principal de l'interpréteur est d'analyser une syntaxe représentant un programme WHILE et de la transformer en AST[1], puis d'exécuter les instructions de cet AST reçu en paramètres sur une mémoire reçue elle aussi en paramètres.

Programme

Structure du programme

Dans le fichier principal `interpreter.scala`, le squelette du projet qui nous a été fourni comporte une structure en plusieurs parties. Les plus importantes sont les parties Analyseur syntaxique et Interpréteur. La partie Interpréteur possède quatre sous-parties : Gestion de la mémoire, traitement des Expressions, des Commandes, et des Programmes.

La partie Analyseur syntaxique sert principalement à analyser et transformer une chaîne de caractères représentant un programme ou une partie d'un programme WHILE syntaxiquement correct en un AST, ceci en ne modifiant pas la sémantique de ce programme.

Dans la partie Interpréteur, la sous-partie Gestion de la mémoire a pour principale utilité de définir le type `Memory` et de réaliser des opérations sur les mémoires (recherches de valeur, affectations).

La sous-partie Expressions sert principalement à passer respectivement d'un AST représentant une `Expression` à la valeur correspondante, et d'une valeur à l'AST correspondant représentant une `Expression` .

La sous partie Commandes a quant à elle pour objectif d'exécuter des commandes représentées par un AST sur une mémoire, et de rendre la mémoire modifiée.

Enfin la partie Programmes utilise les parties précédentes pour réaliser l'interpréteur qui à partir d'un programme et d'une liste de valeurs, rend la liste de valeurs modifiée par le programme.

1 AST : Abstract Syntax Tree

Méthodes

Les méthodes de l'application sont ainsi départagées entre les différentes sections listées ci-dessus.

Comme nous l'avons vu, la logique applicative du programme se concentre dans la section Programmes. La méthode principale de l'application, `interpreter()`, s'y trouve donc. Le reste du programme se compose en grande partie de méthodes auxiliaires nécessaires à la réalisation de cette méthode principale, respectant ainsi certaines préconisations de génie logiciel.

`interpreter()` est donc la méthode racine qui exécute lorsque c'est nécessaire chaque partie du programme.

Difficultés rencontrées

Ce qui nous a le plus causé de difficultés lors de l'implémentation des spécifications fournies, c'est la partie traitement des commandes du langage WHILE, et notamment la méthode `interpreterCommand()`. Nous avons en effet eu quelques difficultés à trouver les solutions adéquates pour les cas `While` et `For` du pattern matching. La consultation du cours s'est alors avérée indispensable.

Conclusion

La réalisation de ce projet nous a permis de mettre en pratique le cours de calculabilité, et de compléter la première partie du pretty printer que nous avions réalisé au cours des séances précédentes. Nous avons ainsi une meilleure compréhension de certains des processus intervenant lors de l'interprétation et la compilation d'un programme informatique, bien que nous restions encore concentré à un haut niveau.