MORRA-TAROUDJIT Jordan – CHEUNG Clément Phelma SEI 2016-2017

Projet Informatique S3 : Interpréteur SCHEME

Rapport de second livrable
17 Octobre 2016

I – Contexte & Objectifs

Il s'agit pour ce deuxième livrable de tout d'abord implémenter la fonction « eval » permettant d'évaluer les formes « define », « set! », « if », « quote », « and » et « or » ainsi que les objets auto-évaluant (atomes) puis de gérer la création et gestion de plusieurs environnements SCHEME.

Compte tenu de notre avancement concernant le livrable 1, il a aussi été question pour nous de le terminer.

II – Démarche Suivie

La finalisation du livrable 1 nous a conduit à la création de 2 types d'objet supplémentaires :

- « make_arith_op » qui définit un objet de type opération arithmétique prenant pour l'instant en charge des opérations simple (*,/,+,-) mais il sera complété à l'issu du livrable 3.

- « make_special_atom » qui définit les atomes ayant une fonction spéciale (@, %, ^, ...) mais qui pour le moment non pas vocation à être utilisés dans notre interpréteur pour l'instant.

Nous avons aussi créé une fonction « sfs_print_expr » qui se charge de l'affichage de l'expression saisie ou de sa valeur après évaluation. Elle regroupe la fonction « sfs_print » plus toute la mise en forme des expressions (⇒, ouverture 1ère parenthèse, retour à la ligne après affichage). Bien entendu les fonctions créées lors du premier livrable ont subit quelques modifications.

Une fois les tests du premier livrable passés, nous avons pu nous pencher sur la réalisation de la fonction « eval » ainsi que des formes que nous devions reconnaître. Pour cela, nous créons d'abord des variables (objets) globaux correspondant aux formes que nous devons implémenter et qui sont déclarées lors de l'initialisation de l'interpréteur. Ensuite, la fonction « read_atom » doit les reconnaître mais aussi repérer l'écriture contractée « ' » de la forme « quote ». Cette partie de transformation de *'a* en *(quote a)* a été gérée dans read.c au niveau de la lecture des symboles, elle retourne une chaine de caractères modifiée en adéquation.

Enfin, la fonction « eval » s'occupe en fonction des formes contenues dans la S-expression de l'évaluer c'est-à-dire :
- la création de variables dans un environnement si « define » est trouvée
- la modification de la valeur d'une variable avec « set ! »
- l'affichage de ce qui suit la forme « quote »
- l'exécution d'une condition avec « if », évaluation de la conséquence ou de l'alternative
- l'exécution des opérateurs « and » et « or »

Dans le même temps, nous avons réfléchit à l'implémentation des environnements scheme. Deux-trois solutions sont possibles :
- la première consiste à réaliser un tableau de pointeur vers des listes contenant toutes les variables rattachées à un environnement
- la deuxième consiste à réutiliser les paires qui sont déjà implémentées pour réaliser un arbre des environnement contenant aussi un arbre des variables qui lui sont propres. Tous ceci constitue un arbre de paires dont les car sont les environnement et les cdr sont l'arbre des variables.
- la troisième solution consiste à réaliser une table de hachage.

Nous avons décidé dans un premier temps de réaliser la deuxième méthode. Nous verrons par la suite si nous tenterons d'implémenter une table de hachage qui est une méthode plus optimale.

III – Résultats Obtenus & Evaluation

Les fonctionnalités du premier incrément sont enfin réalisées et fonctionnent. Nous sommes donc en mesure de créer l'arbre syntaxique, d'afficher la S-expression, et de détecter des erreurs de frappes sur les atomes.

Pour ce deuxième incrément, seule la fonction d'évaluation est implémentée mais elle doit être plus rigoureusement testée afin de valider son bon fonctionnement . Nous n'avons pas eu le temps en revanche de réaliser la création et la gestion des environnements vu que nous accusons un retard assez important mais nous le rattrapons petit à petit.

IV – Suite du projet

Comme dit précédemment, nous devons tester de façon plus intensive la fonction « eval » ainsi qu'implémenter les environnements scheme et le tester. Des optimisations de performance mais aussi de clarté / propreté du code seront à prévoir notamment au niveau de l'utilisation des ressources mémoires. Je rappelle que pour le moment le champ string des objets à une taille égale à BIGSTRING soit plus de 65000 caractères, ce qui est dans la grande majorité des cas disproportionné et donc très loin d'être optimal.