SORBONNE UNIVERSITÉ

Analyse des Programmes et Sémantique

Rapport de projet

Léa Delannay

État d'avancement :

Les exécutables prologTerm, eval et typrog sont entièrement fonctionnels pour les versions APS0 et APS1.
Les tests ont été réalisés à l'aide de fichiers trouvés sur le site 2017-2018 de l'UE. Certains ne passent pas mais cela est normal car ils étaient soit mal typés, soit pas adaptés à notre version d'APS qui est un peu différente des années passées.

Je me suis arrêtée à la version APS1a, pour celle si prologTerm est fonctionnel, le typeur ne marche pas. Je n'ai pas réussi à résoudre mes erreurs je pense que c'est en rapport avec ref dans le contexte. Je n'ai pas eu le temps de me consacrer à l'évaluateur.

Choix d'implémentation :

J'ai choisi d'implémenter tout ce qui est ast.ml, lexer.mll, parser.mly en OCAML car c'était le langage que je trouvais le plus intuitif par rapport au pattern matching. De plus en java par exemple il faut créer énormément de fichier, ce qui n'est pas le cas ici où un seul fichier suffit pour chaque composant.
Ensuite le typer a été fait en prolog, et l'évaluateur ainsi que le prologTerm en OCAML aussi.

Pour APS0, il n'y a pas de choix d'implémentation spécifique, mis à part le fait que dans ast.ml j'ai choisi de ne mettre que les types au singulier. C'est-à-dire que par exemple pour les *sexpr* il n'y a que le type « *sexpr* » et dans le parser.mly pour s*exprs* j'ai choisi de le déclarer comme « *<Ast.sexpr> sexprs* ».
De plus dans le lexer j'ai décidé de différencier les opérateurs *add, mult*(…) avec le if et le not. Mais pout *add,mult* etc j'ai tout regroupé dans le mot PRIM, qui va contenir une string. Et c'est comme ça que je pourrais récupérer le nom de l'opérateur, je trouve ça plus optimisé que de créer un token pour chaque opérateur. De plus il n'y a donc pas besoin de créer un type de chaque dans le fichier de l'ast.
Pour l'évaluateur j'ai décidé de considérer l'environnement comme une liste de clé/valeur, je peux donc utiliser les fonctions qui les manipule. Je la donne vide au début avec eval_prog. Et j'ai décidé de ne pas tenir une liste pour la sortie, en effet à chaque appel de *echo* je print directement au lieu de tenir une liste et de la print à la fin.

A partir de APS1 j'ai essayé de lever des exceptions dans l'évaluateur, chose que je n'ai pas faite pour APS0.
Par ailleurs, les choix d'implémentations restent cohérents avec la version ultérieure. La mémoire est modélisée aussi par une liste de clé/valeur.

Pour APS1a, il a fallu rajouter « exprp » et « *argp* », je n'ai pas voulu rajouter de type dans l'ast je les ai directement inclus dans *sexpr* et *arg* respectivement.
J'ai fait cela car on peut remarquer qu'une *exprp* peut être une *sexpr* et un *argp* peut être un arg.
Ainsi pour prologTerm, il n'y avait que quelques lignes à rajouter pour matcher les 2 nouveaux AST.