Rapport Projet TAS - Implémentation d'un langage de programmation avec typage

Typeur, Evaluateur , Parser

Dibassi Brahima 21210230

2023-11-19

Table des matières

1 Langage de programmation choisi

Afin d'implémenter ce projet, nous avons choisi le langage de programmation *OCaml*. Ayant déjà expérimenté l'implémentation de l'interpréteur d'un langage en OCaml suite au cours d' **APS**, nous avons pu constater que ce langage était très adapté à ce genre de projet.
Grâce à la somme de produits au pattern-matching associé, et au support des paradigmes, imperatif et fonctionnel, nous avons pu nous servir des différents traits du langage afin de faciliter notre implémentation de l'évaluateur et du typeur.

2 Structure du projet

- Repertoire `Lib/`
  - `lexer.mll` : Création des tokens
  - `parser.mly` : Parser afin de construire notre arbre syntaxique abstrait depuis les tokens
  - `ast.ml` : Structure de l'AST du langage avec les types des expressions
  - `baselib.ml` : Définitions des primitives du langage et leur types
  - `evaluator.ml` : Partie sémentique du langage, évaluation des expressions
  - `prettyprinter.ml` : Fonctions d'affichage
  - `typeur.ml` : Partie analyse statique, vérification des types
  - `typingEnv.ml` : Gestion de l'environnement des déclarations de types par l'utilisateur
- Repertoire `Bin/`
  - `main.ml` : Contient notre script principal, lancé lors de la commande `dune exec ProjetTAS *fichier*.ml`. Il vérifie si le programme .ml ciblé est bien typé et l'évalue si c'est le cas en faisant tous les affichages sur le terminal. Nous avons choisi d'afficher également l'état de la mémoire à la fin de l'évaluation du programme.
- Repertoire `Test/`
  - `testing.ml` : Script de test, permet de tester tous les fichiers .ml présents dans le dossier `test` et de vérifier si le typeur et l'évaluateur fonctionnent correctement.
  - `yamlHelper.ml` : Fonctions d'aide pour la lecture des fichiers de test au format yaml
  - `template.yaml` : Template de fichier de test au format yaml

Les tests sont dans le dossier `test` classés en fonction des extentions du langage auxquels ils correspondent. Les extensions du langage devant également passer les tests des versions précédentes.

3 Syntaxe du langage

4 Difficultés rencontrées

5 Extensions

5.1 User-Defined types

5.2 MatchPattern

5.3 Annotations de types

5.4 Gestion des erreurs