Sorbonne Univeristé
Faculté des Sciences et Ingénierie
Département d'Informatique

Rapport projet APS

Younes CHETOUANI & Malek BOUZARKOUNA

Mai 2024

M1 STL **MU4IN503**

1 Introduction

Dans le cadre de l'UE d'APS, nous devions implémenter les versions 0 à 2 du langage APS. Pour y parvenir, nous avons alors décidé de suivre le conseil de Mr.Demangeon et de le réaliser en Ocaml.

Ainsi, pour chaque version il nous revenait d'implémenter l'analyse syntaxique du langage (lexer + parser), le typage du langage (cette partie est imposée en Prolog) et enfin l'évaluation.

Les différents programmes APS utilisés pour tester notre implémentation nous ont été très gracieusement partagés par le binôme de Yanis et Salim Tabellout. Ils sont tous placés dans les dossiers `/Samples` des différentes versions d'APS. Dans le dossier `/Expected` se trouvent les résultats attendus lors du test de tyapage et de l'évaluation des différents programmes test. Afin de vous faciliter la tâche nous avons intégré un script permettant d'éxécuter le test d'évaluation et de typage de tous les programmes test.

Etat d'avancement : Nous avons implémenté en entier les versions d'APS0 à APS1a et partiellement APS2. Seuls le lexer, parser et typeur d'APS2 ont été implémentés. L'évaluateur d'APS2 n'est pas fonctionnel.

2 Manuel utilisateur

Chaque version d'APS implémentée a son code source stocké dans un dossier nominatif. Chacun de ses dossiers est structuré de cette façon :

```
|-- Aps0
|   |-- Expected
|       |-- abs_1
|           |-- eval.result
|           |-- type.result
|       |-- ...
|   |-- Samples
|       |-- abs_1.aps
|       |-- ...
|   |-- Makefile
|   |-- ast.ml
|   |-- eval.ml
|   |-- lexer.mll
|   |-- parser.mly
|   |-- prologTerm.ml
|   |-- test.sh
|   |-- type_checker.pl
|   |-- typrog
|-- Aps1
|-- Aps1a
|-- Aps2
|-- readme.md
```

FIGURE 1 – Arborescence des fichiers

L'ensemble des fichiers APS à tester sont dans le répertoire `/Samples` et les résultats attendu lors de leur typage et évaluation sont stockés dans le répertoire `/Expected`. Par exemple, les résultats attendus pour le fichier add.aps sont dans le sous-dossier add du répertoire `/Expected`. Le résultat du typage attendu est contenu dans le fichier `type.result` et celui de l'évaluation dans `eval.result`. En ce qui concerne l'éxécution des tests veuillez vous référer au guide d'utilisation du projet (`readme.md`).

3 Portée du projet

Il est difficile d'assurer que l'implémentation est entièrement fonctionnelle car cela requerrait une base de test encore plus exhaustive que celle que nous proposons. Cependant nous avons tout de même essayé de tester l'ensemble des fonctionnalités de chaque versions d'APS et tous les tests passent en étant correctement typés d'APS0 à APS2 et évalués d'APS0 à APS1a.
De plus, nous avons vérifié qu'il y ai rétrocompatibilité à chaque nouvelle version d'APS en retestant les programmes des versions précédentes.

Ainsi, nos tests nous laissent penser que notre implémentation du langage APS est entièrement correcte d'APS0 à APS1a et correcte pour le typage d'APS2.

4 Choix d'implémentation et difficultés rencontrées

Nous avons choisi d'utiliser Ocaml en suivant le modèle fourni, ce qui nous a donné une base solide pour commencer à travailler sur notre projet. Cependant, nous avons rencontré plusieurs défis tout au long du processus d'implémentation.

Tout d'abord, une part importante de notre temps a été consacrée à la compréhension du fonctionnement de Prolog. Nous avons dû nous familiariser avec ses concepts uniques tels que les relations, les règles et les backtracking. Cela a entraîné un retard significatif dans notre travail sur APS0.

Le plus grand défi est survenu lors du débogage du typeur Prolog. L'utilisation de la commande `trace` pour suivre le typage du programme et identifier les erreurs s'est avérée particulièrement pénible. Les traces produites lors de l'exécution étaient denses et parfois presque illisibles, ce qui rendait difficile de localiser précisément où le typage posait problème. C'est pourquoi le débogage des typeurs a pris plus de temps que prévu.

Malgré ces difficultés rencontrées sur APS0, nous avons été en mesure de surmonter ces obstacles et de continuer à progresser dans notre projet. Une fois APS0 implémenté, le reste s'est déroulé sans trop d'encombres. Les bases solides que nous avons posées au début du projet ont porté leurs fruits, et nous avons pu avancer plus efficacement par la suite.