Rapport d'activité PCL

BIAUSQUE Anna
IORI-GINGEMBRE Nathan
PASQUIER Pierre

16 Janvier 2023

Table des matières

Chapitre 1

Introduction

Cette dernière partie du rapport de PCL va être ciblée sur la création de la table des symboles et les controles sémantiques qui peuvent en découler. La table des symboles a pour objectif de synthétiser la déclaration de nouvelles variables ainsi que de hiérarchiser un programme écrit en TIGER à l'aide de numéros de région et de numéros d'imbrication pour chacune de ses tables. Grâce à cette table des symboles et à plusieurs fonctions permettant d'en faire ressortir les informations, il nous est par la suite possible de mettre en place des contrôles sémantiques qui vont avoir pour but de détecter les potentielles erreurs de cohérence qui se glisseraient dans notre programme tel que les erreurs d'appel de fonctions avant leurs créations ou encore des erreurs de typages.

Chapitre 2

Table des symboles

Avant de construire notre table des symboles, nous avons dû choisir une structure pour que les données puissent être retrouvées facilement. Tout d'abord, chaque table contient un nom sous la forme TDS_Nom_NuméroRégion _NuméroImbrication. De plus, chaque ligne de chaque table représente les informations d'un symbole. Ces lignes commencent par le nom de ce symbole, sont suivies de son attribut et du reste des informations en fonction de cet attribut. Voilà les informations que l'on stocke en fonction des attributs :

- VAR : défini une variable et est suivi du type, nombre d'éléments, taille de la matrice si s'en est une et déplacement
- TYPEID : défini une déclaration de type avec un simple identifiant et est suivi de cet identifiant
- TYPEFIELD : défni la déclaration de type pour les "records"/"vecteurs" et est suivi du nom et du type de chaque élément du vecteur
- TYPEARRAY : défini la déclaration de type pour les matrices et est suivi du type des éléments dans la matrice
- METHOD : défini une méthode et est suivi du type de retour de la méthode, de son nombre d'arguments et du type de chaque argument
- PARAM : défni un paramètre de méthode et est suivi du type de ce paramètre et de son déplacement
- COMPT : défni un compteur de boucle et est suivi des bornes inférieures et supérieures de compteur

Voilà un exemple de ce qu'on peut obtenir sur le programme des 8 reines :

```
TDS_let_1_0 :
[N, VAR]
[intArray, TYPEARRAY, int]
[row, VAR, intArray, 0]
[col, VAR, intArray, 0]
[diag1, VAR, intArray, N+N-1, 0]
[diag2, VAR, intArray, N+N-1, 0]
[try, METHOD, void, 1, int]

TDS_printboard_2_1 :
[c, PARAM, int]
[a, PARAM, string, int]
[printboard, METHOD, int, 2, int, string]

TDS_for_3_2 :
[i, COMPT, 0, N-1]

TDS_for_4_3 :
[j, COMPT, 0, N*1+1]

TDS_if_5_4 :

TDS_try_6_1 :
[c, PARAM, int]

TDS_if_7_2 :

TDS_for_8_3 :
[r, COMPT, 0, N-1]

TDS_if_9_4 :
```

FIGURE 2.1 – Exemple TDS

Chapitre 3

Contrôles sémantiques

Nous avons décidé de faire nos contrôles sémantiques dans les classes que nous avions définies pour faire l'arbre abstrait. Pour cela, nous avons défini une fonction ControleSemantique() est dans chaque classe. Ainsi, chaque nœud de l'arbre appelle la fonction ControleSemantique de ses fils qui lui-même appelle celle de ses fils ... Ce qui permet de parcourir tous les nœuds de l'arbre. Il ne nous reste plus qu'à faire remonter les bonnes informations des feuilles de l'arbre jusqu'à l'endroit où on effectue les contrôles. Par exemple, les identifiants, qui forment toujours une feuille de l'arbre font remonter leur type (s'ils sont définis dans la TDS, sinon on affiche une erreur) ainsi que leur nom ce qui permet de comparer le type des 2 opérandes dans les calculs ou les comparaisons. Le problème que nous avons rencontré était d'obtenir le numéro de ligne, les numéros de région et d'imbrication ainsi que la table des symboles, car ces variables n'étaient pas définies dans les classes, mais seulement dans l'AstCreator. Pour résoudre ce problème, nous avons passé ces variables en arguments des constructeurs de chaque nœud ce qui nous a permis d'y avoir accès dans les classes.

Chapitre 4

Tests

Pour les tests de la table des symboles, une fois que nous nous étions mis d'accord sur quels éléments allait former des nouvelles tables et comment allaient être rangés les éléments dans ces tables nous nous sommes servis de programmes simples afin de pouvoir voir comment les tables se formaient et si c'était bien cohérent vis à vis de nos attentes puis nous avons réglé les derniers problèmes restant en essayant sur des programmes plus longs et complexes.

Pour les tests des contrôles sémantiques, nous sommes partis du programme des 8 reines correct et nous avons simulé les erreurs que l'on voulait être en capacité de détecter. Cela nous a permis de tester les erreurs une à une, mais aussi d'en tester plusieurs à la suite pour s'assurer que la détection d'une erreur ne bloque pas la détection des erreurs dans la suite du programme.

Chapitre 5

Gestion de projet

Pour chaque partie, nous nous sommes réparti le travail à faire et avons fait des réunions régulièrement pour voir si tout le monde avançait bien et aider ceux qui en avaient besoin. Voici les tâches que chacun a réalisées pour la table des symboles :

Pierre :

— Création des nouvelles tables
— Déclaration des types et des fonctions
— Fonctions pour ajouter des calculs à la table

Nathan :

— Ajout des compteurs des boucles for

Anna :

— Déclaration de variables

Et voici le travail réalisé pour les contrôles sémantiques :

Pierre :

— Fonctions pour récupérer les informations utiles dans la table des symboles
— Mise en place de la circulation des informations dans l'arbre pour effectuer les contrôles de type
— Mise en place de la récupération des variables nécessaires aux contrôles de déclaration et portée des variables
— Contrôle du nombre d'arguments dans les appels de fonction
— Contrôle du type des arguments dans les appels de fonction
— Contrôle du type des opérandes pour la multiplication, addition, division, soustraction
— Contrôle du type de retour des fonctions
— Contrôle des fonctions bien déclarées avant utilisation
— Contrôle des variables bien déclarées avant utilisation

Nathan :

— Contrôle de la déclaration de variables avec le bon type
— Contrôle du type des conditions dans le ifthenelse et le while
— Contrôle du type des bornes d'une boucle For
— Contrôle du type de déclaration d'une liste et des éléments la composant
— Contrôle de l'affectation de variables avec le bon type

Étant donné qu'Anna n'a pas pu finir le projet pour raison médicale, nous n'avons pas les contrôles liés aux déclarations de variable hors mis une simple vérification que la variable a déjà été déclarée, sans tenir compte de sa visibilité. Nous avons tout de même fait en sorte que ces contrôles soient implémentables en s'assurant d'avoir toutes les variables nécessaires à ces derniers-là, il aurait fallu les réaliser.