Rapport d'activité 2 (Semaine 49 06/12/2021)

1. Gestion de projet

Ce projet est réalisé par les membres présentés dans le tableau ci-dessous. La méthode de travail [1] du précédent rapport d'activité est conservée. Le suivi des tâches se fait avec le Trello [2] et les compte-rendus de réunions sont déposés sur Overleaf et sur le dépôt `gitlab`. La répartition des tâches au sein du groupe durant cette période est détaillée dans les paragraphes suivants.

Membres de l'équipe	Rôles ou charges	Temps de travail
Chaima TOUNSI-OMEZZINE	Chef de projet	7h
Céline ZHANG	Secrétaire	7h
Gisel RODRIGUEZ-BAIDE	Reviewer	8h

Chaima TOUNSI-OMEZZINE a continué l'écriture des tests, a produit une partie des visiteurs GraphViz pour générer la représentation graphique de l'AST et a debug du code.

Céline ZHANG a complété les visiteurs de l'AST, a corrigé la grammaire, a vérifié les tests de l'AST, a revu et debug les codes.

Gisel RODRIGUEZ BAIDE a produit les visiteurs pour générer l'AST et a écrit une partie des visiteurs GraphViz pour générer la représentation graphique de l'AST.

2. Travail effectué

2.1. Grammaire

Une deuxième version de la grammaire CIR-C a été réalisée [3]. Par rapport à la première version, quelques règles ont été modifiées pour faciliter l'écriture des visiteurs de l'AST et pour être cohérent sur les productions possibles. Par exemple, pour les affectations, il faut avoir la possibilité d'alterner les variables `a = b->c = u = d->y = 2`. De plus, on a utilisé des labels sur la grande majorité des sous-règles de la grammaire pour différencier les cas qui doivent être traités séparément lors de la génération de l'AST.

2.2. AST

Pour générer l'AST, nous avons appliqué les méthodes vue lors du TP2 de PCL : l'implémentation du patron de conception Visiteur. Pour cela, à partir du `Parser` et des `BaseVisitors` produites lors de la compilation, nous avons tout d'abord créé une interface AstVisitor qui contient tous les visiteurs de notre AST, une classe AstCreator qui, pour chaque classe de notre AST, implémente la méthode visit. Cette même classe étend exprBaseVisitor<Ast>. De plus, pour chaque noeud visité, une méthode accept et des sous-classes de AST sont créées (si nécessaire).

En ce qui concerne la représentation graphique de l'AST, on a utilisé les outils du package graphViz dans la classe GraphVizVisitor, dans laquelle se trouvent tous les visiteurs associés qui ont été contruits pour visiter l'arbre syntaxique pûr dans le but de construire l'arbre syntaxique abstrait.

1. une méthode de travail classique avec des réunions hebdomadaires et stand-up meetings si besoin
2. `https://trello.com/b/E3EJnI0p/projetpcl-1`
3. elle se trouve dans le fichier `grammaire.g4` dans notre dépôt gitlab

2.3. Tests

Plusieurs fichiers de tests exhaustifs ont été rédigés afin de tester la génération de notre AST. Les déclarations de variables, de fonctions, de structures, les blocs d'instructions, la priorité des opérations et la gestion des commentaires ont été traitées dans ces tests.
Ainsi, ci-dessous, se trouve un exemple de test et l'AST correspondant.

```
int func(int x, int y) {
  int n;    // if (chaima==true) gisel==true; else celine==false;
  u888 = t->n = x + 'c' + '\'' + fct_void() + sizeof(struct st);return n;    /* if (
   test==20) celine==happy else celine==sade;
                      giufheiuyhfuyzegfpiu*/
  /* t->r->a->d = sizeof(struct st); affectation ->-> pose probleme */
}
// while (test=passed) we_are_happy; /* hellloooooo */ if then else ;
/* /// //* fefefe /* fefef */
struct oneStruct {
  int age, taille, poids;
  struct anotherStruct *root, *child;
};
```

Listing 1 – Test

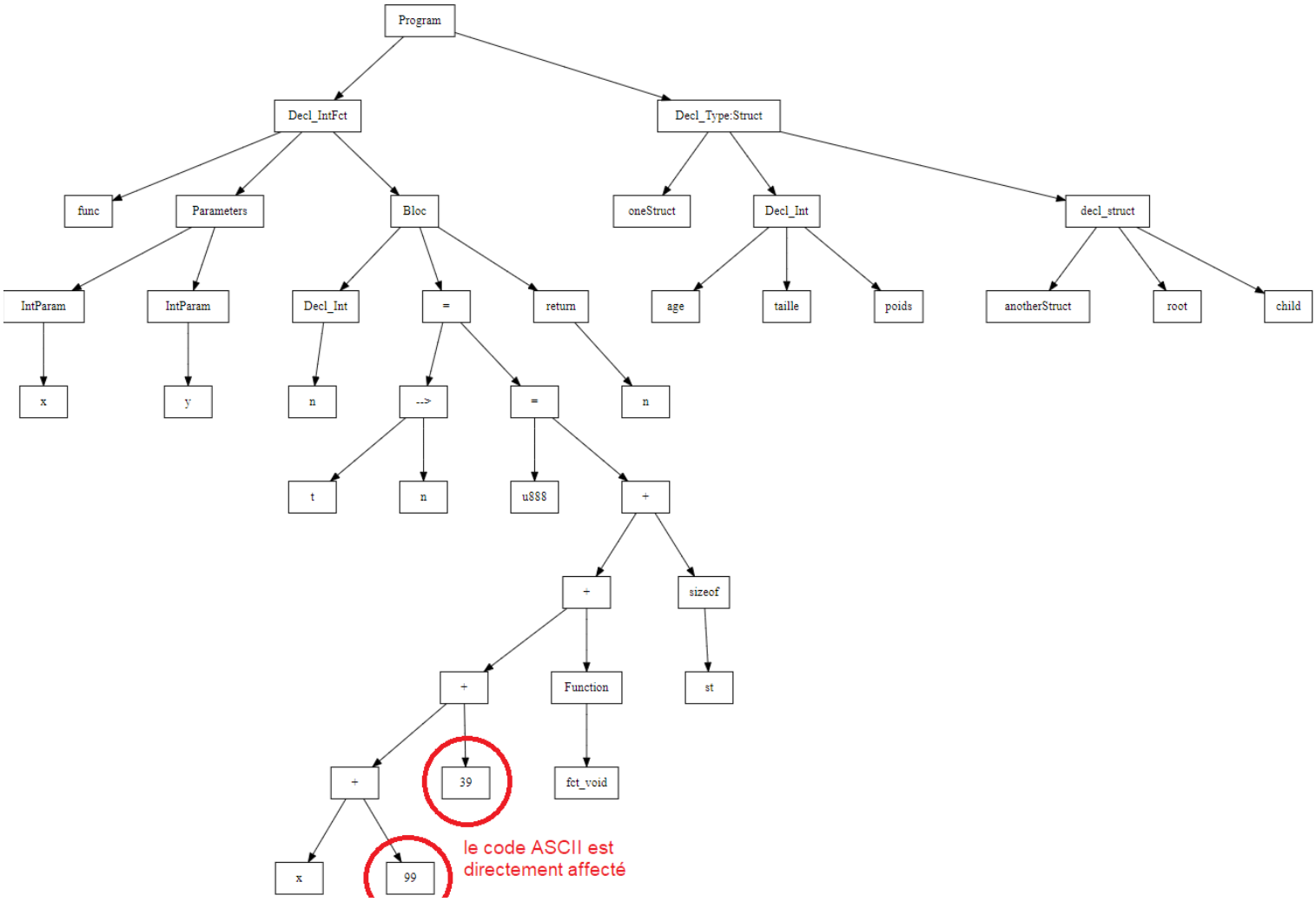

FIGURE 1 – AST du code Test

3. Problèmes rencontrés et solutions

3.1. Grammaire

Dans la grammaire actuelle, la règle `fleche` pose certainement un problème lors de la reconnaissance de plusieurs flèches : `a->b->c`. La cause n'a pas été réellement identifiée, ce soucis est apparu lors de la création de L'AST, donc des hypothèses sont émises. Soit le problème provient des visiteurs de GraphVizVisitor, soit il provient des visiteurs de l'AST, soit il provient de la grammaire (moins probable).

3.2. AST

En lançant les tests, des problèmes dans la génération de l'AST ont été relevés. C'était majoritairement, des erreurs d'inattention dans l'implémentation des visiteurs qui ont pu être corrigées. Cependant, des erreurs comme le traitement des fonctions qui n'ont pas de paramètres ont été détectées et résolues après quelques modifications des visiteurs correspondants.

Par ailleurs, nous avons eu des problèmes avec la représentation des `char` lors de la production de l'AST, en effet, nous traitons les `Integer` comme des entiers, et récupérer un `'c'` n'a pas plus au visiteur. De ce fait, un traitement supplémentaire a été ajouté pour récupérer le code ASCII directement.

D'autres problèmes comme la création du noeud `return` ou `sizeof` ont été réglés, car précédemment, il n'était pas possible de reconnaître que la feuille issue de `instructions` était un `return N;` et non simplement un `N;`. Pour régler cela, nous avons rajouté un nœud "return" comme parent de chaque valeur retournée, et un nœud `sizeof` en tant que père du `IDF` passé en paramètre de `sizeof`. Mais, on aurait pu aussi laisser comme père `value` et ajouter une feuille `sizeof`.

4. Travail à venir

Pour le mois à venir, on perfectionnera la génération de l'AST avec les quelques ajustements qui restent à faire et on réalisera l'analyse sémantique de la grammaire avec une table des symboles qui sera construite à cet effet. Pour cela, nous allons écrire les visiteurs permettant de visiter l'AST, puis les classes qui permettent le remplissage de la TDS, par exemple les classes `TDSAttribut` et `TDSFunction` qui hériterons de la classe abstraite `TDSLine` (représentant les lignes de la TDS). Les contrôles sémantiques se feront en parcourant l'AST et la TDS, avec des visiteurs aussi, pour vérifier les déclarations, les types, les valeurs, etc.

TO-DO LIST

- Se renseigner sur les problèmes : OK
- Modifier les visiteurs de l'AST en conséquence des changements de la grammaire
- Réaliser la table des symboles (TDS)
- Réaliser le contrôle sémantique