Telecom Nancy

Projet Compil

Rapport d'activité - Semaine 49

Membres du projet :
ABDOULHOUSSEN Hamza
BOULECHFAR Sami
KHATIB Maha

Responsables du projet :
Collin Suzanne
Da Silva Sébastien
Bourbeillon Alexandre

Table des matières

1 Avancement du travail :

1.1 Grammaire :

- Ajout de terminaux : `OPERATEUR` et `LETTER`.
- Ajout de non terminaux pour gérer la priorité et l'associativité des fils de expr : `egal`, `ou`, `et`, `diff`, `comp`, `plus`, `mult`, `unaire`, `fleche` et `value`

1.2 Ast :

- Création des noeuds d'AST : `Affect.java` , `Bloc.java` , `DeclVarInt.java` , `DeclVarStruct.java` , `Decl_fct_int.java` , `Decl_fct_int_param.java` , `Decl_fct_struct.java` , `Decl_fct_struct_param.java` , `Decl_fct_struct_param_vide.java` , `Decl_typ.java` , `Div.java` , `Egal.java` , `Et.java` , `Fichier.java` , `Fleche.java` , `Ident.java` , `If.java` , `IfElse.java` , `Inegal.java` , `Inf.java` , `InfEgal.java` , `Int.java` , `Moins.java` , `Moinsunaire.java` , `Mult.java` , `Not.java` , `Ou.java` , `ParamInt.java` , `ParamStruct.java` , `Plus.java` , `Return.java` , `Sup.java` , `SupEgal.java` , `Value_expr.java` , `Value_list_expr.java` , `Value_list_expr_vide.java` , `Value_sizeof.java` , `While.java`
- Mise à jour d'AstCreator
  À partir d'`exprBaseVisitor.java` l'équipe a complété les méthodes visit permettant de construire l'AST. Pour les noeuds binaires et unaires correspondant aux différents opérateurs, l'équipe a utilisé des noeuds temporaire pour faire apparaître l'associativité dans l'AST.

  On propose un exemple d'AST avec opérations binaires (figure 1) [1] et un exemple d'AST de structure (figure 2) :

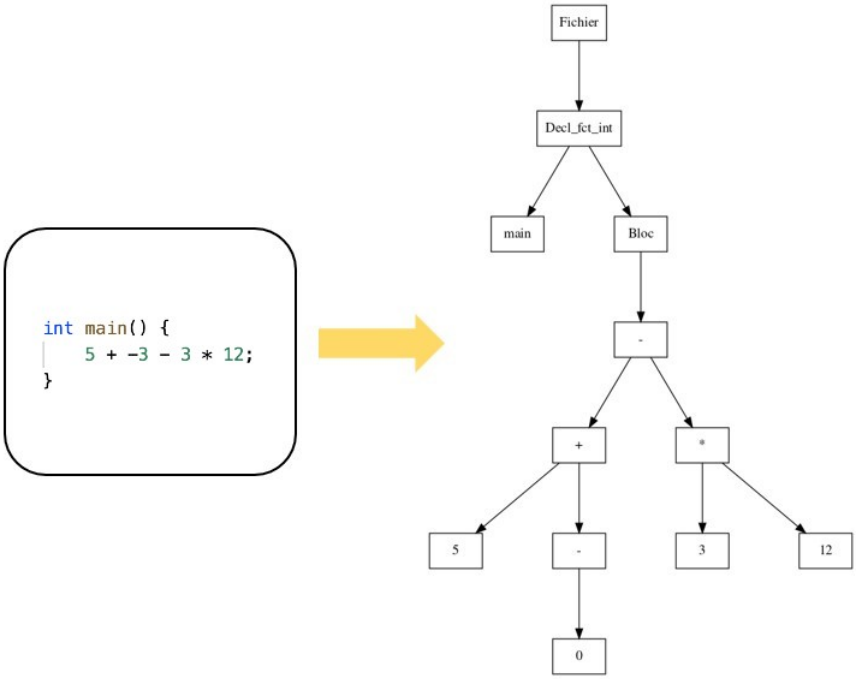

FIGURE 1 – AST avec opérations binaire

1. Cet AST présente des noeuds contenant un moins unaire et un moins binaire (n'ayant pas la même sémantique), il faudra par ailleurs distinguer ces noeuds autrement que par leurs nombres de fils

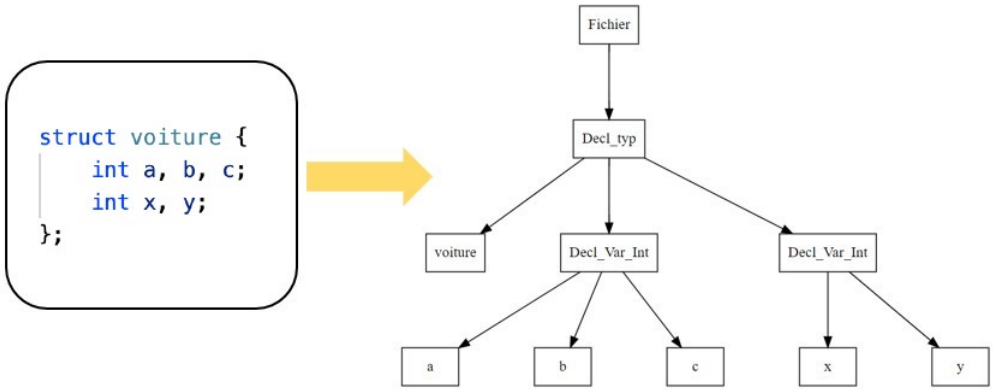

FIGURE 2 – AST d'une structure

- `AstVisitor` et `GraphVizVisitor` L'équipe a complété l'interface `AstVisitor` et a écrit les méthodes visit de `GraphVizVisitor` permettant de générer le fichier dot et ainsi d'avoir une représentation graphique de l'AST. Un exemple d'AST affiché est donné plus haut (figure 1).

1.3 Tests :

- Modification du launch.sh
  Le script `launch.sh` permet maintenant de générer les AST sous forme dot et svg. Il peut être raccourci en ajoutant des paramètres. Toutes les précisions sur l'exécution du code sont données dans le `README.md` du dépôt git (`https://gitlab.telecomnancy.univ-lorraine.fr/Sami.Boulechfar/khatib3u`)

- Création d'arbre sur tous les tests
  Le script `launchalltree.sh` dans le dossier `bash` permet de générer les AST pour tous les fichiers tests. Les membres vérifient ainsi la génération d'AST pour tous les tests en une commande.

2 Les difficultés rencontrées :

2.1 Priorisation et associativité de la grammaire :

La grammaire qu'on avait auparavant ne permettait pas de gérer la priorité de nos opérateurs ainsi que leur associativité. Pour ce faire, on s'est basé sur ce qui a été vu en module de MI l'année dernière ainsi que ce qui a été fait en TP compil cette année. L'objectif était de se retrouver avec un arbre comportant les opérateurs les plus prioritaires en bas. Il fallait également faire attention à l'associativité à droite de l'opérateur =, contrairement aux autres opérateurs ayant une associativité à gauche. Ainsi, on a pu établir les relations de précédence et d'associativité des opérateurs à travers le tableau suivant, allant de la plus faible à la plus forte précédence :

opérateur	associativité	précédence
=	à droite	plus faible
||	à gauche	
&&	à gauche	
== !=	à gauche	
< <= > >=	à gauche	
+ −	à gauche	
* /	à gauche	
! − (unaire)	à gauche	
−>	à gauche	plus forte


```
expr : egal ;

egal : (ou '=')* ou;
ou : et ('||' et)* ;
et : diff ('&&' diff)*;
diff : comp (('==' | '!=')comp)* ;
comp : plus (('<' | '<=' | '>' | '>=')plus)
plus : mult (('+' | '-')mult)* ;
mult : unaire (('*' | '/')unaire)* ;
unaire : ('!' | '-')* fleche ;
fleche : value ('->' IDENT)* ;
```

FIGURE 3 – Du tableau des priorités à la grammaire

2.2 Analyse de la grammaire :

- La grammaire ci-dessous ne parvenait pas à identifier un `INT` pour un seul chiffre autre que `0`.

```
CHIFFRE : ('0'..'9');

INT : '0' | ('1'..'9') CHIFFRE* | LETTER ;

LETTER : 'A'..'Z' | 'a'..'z' | '_';
```

  Le problème a été résolu en inversant l'ordre de `CHIFFRE` et `INT`.

2.3 Génération d'AST :

- Compréhension et correction de la classe GraphVizVisitor permetant d'écrire un fichier dot.
  Le fichier dot généré ne compilait pas, nous avons ajouté un `endBuffer` rajoutant une accolade et un retour de ligne dans `buffer` avant de l'écrire dans `output`.

3 Prochaines étapes :

- Ignorer les commentaires
- Vérifier l'AST :
  - Multiplier le nombre de tests pour tester chaque noeud.
  - Modifier le nom des noeuds pour que ce soit clair et correspond à la sémantique.
    - Différencier le moins binaire du moins unaire
    - Dans IfElse, différencier les blocs if et else de tout les autres blocs.
- Parcourir l'arbre pour faire la table des symboles.
  - Réfléchir à la structure utilisée pour la table des symboles
  - Réfléchir aux symboles à placés dans la table
  - Réfléchir au parcours pour repérer les différentes régions
- Faire l'analyse sémantique
- Générer la chaîne de code

4 Gestion de projet :

4.1 Tâches réalisées :

L'équipe s'est réparti les tâches comme ci-dessous :

- Amélioration de la Grammaire : Maha Khatib
- Génération des AST : Sami Boulechfar et Hamza Abdoulhoussen et Maha KHATIB
- Ecriture des tests : Hamza Abdoulhoussen

4.2 Réunions de travail :

4.2.1 Mercredi 24 novembre 2021 :

Présent	Heure	Lieu
Maha KHATIB Sami BOULECHFAR Hamza ABDOULHOUSSEN	15h	TELECOM Nancy

Travail réalisé
L'équipe à d'abord fait le point sur la gestion des priorités et de l'associativité de la grammaire, l'un des membres s'en est chargé et a partagé son travail aux autres. Le groupe a également repris le tp2 de Trad pour réapprendre à coder les noeuds des AST et des premiers noeuds ont été ajouté au dépôt GIT.

Objectifs suivants

- Finir l'écriture des noeuds des AST
- Tester chaque noeud de l'AST
- Aboutir a une version presque finale de la grammaire

4.2.2 Mercredi 1er décembre 2021 :

Présent	Heure	Lieu
Maha KHATIB Sami BOULECHFAR Hamza ABDOULHOUSSEN	14h	TELECOM Nancy

Travail réalisé
L'équipe s'est réuni pour finir l'écriture des noeuds des AST. Elle a ensuite mis en commun tout les noeuds puis a corrigé les différents problèmes rencontrés lors de la génération d'AST avec le code. En parallèle, le groupe a commencé à rédiger le rapport d'activité. L'objectif était d'avancer un maximum sur le projet afin que les membres puissent profiter des semaines à venir pour la préparation des partiels.

Objectifs suivants

- Finir la rédaction du rapport d'activité
- Réaliser plus de test pour vérifier l'AST obtenu sur tout les noeuds
- Commencer brièvement à réfléchir à différents tests sémantiques