Telecom Nancy

Projet Compil

Rapport d'activité - Lundi 10 janvier

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 et modification de terminaux : `COMMENT` et `CARACTERE`.

  Le terminal `COMMENT` permet la gestion des commentaires ainsi que les retours à la ligne et les retours chariots. Par ailleurs, les caractères spéciaux ont été ajoutés dans le terminal `CARACTERE`.
- Modification de labels : Afin de faciliter la compréhension des Ast, l'équipe a changé les noms des labels.

1.2 AST :

Le graphViz ainsi que les classes java ont été modifés en accord avec les nouveaux labels définis pour la grammaire.

1.3 Jeux d'essais :

L'équipe a d'abord réalisé d'avantage de jeux d'essais pour tester la grammaire. Ceux-ci sont répartis dans 5 dossiers :

- Test_Unaire : Chaque test est destiné à évaluer une règle de la grammaire.
- Test_Complet : Chaque test est issu d'ancien code C adapté pour pouvoir être lu par la grammaire. Ces tests sont censés brasser plusieurs règles.
- Test_Erreur : Chaque test présente volontairement une ou plusieurs erreurs syntaxique. Ces tests ne sont pas censés pouvoir être lu par la grammaire.
- Test_Semantique : Chaque test vise à vérifier le bon remplissage de la TDS.
- Test_Erreur_Semantique : Chaque test présente volontairement une ou plusieurs erreurs sémantique. Ces tests ne sont pas censés pouvoir générer une TDS.

1.4 Launch :

Pour lancer les tests, de nouveaux launch sur Linux ont été réalisés. Le dépôt git présente un dossier `bash` dans lequel se trouve tous les launchs appelé par le `launch.sh`. Les launchs pouvaient déjà créer les arbres et les asts. Désormais ils peuvent également créer la TDS sous forme dot et svg. A cela s'ajoute la possibilité de tester en une commande, pour tous les fichiers, la bonne réalisation des arbres, des asts ou des tds. Les launchs permettent également de vérifier les erreurs syntaxiques ou les erreurs sémantiques.

Deux membres du groupe étant sur Windows, l'équipe à ensuite décidé d'adapter ces launch sur Windows. Le dépôt git présente un dossier `Windows` dans lequel se trouve tous les launchs appelé par le `launchW.sh`. Chaque launch peut désormais être lancé depuis Powershell.

1.5 TDS :

L'affichage de la table des symboles se fait en complétant un fichier dot qui est ensuite converti en svg comme pour les AST. Chaque noeud de l'AST est parcouru avec la méthode des visiteurs et la TDS est complétée lors du parcours.

1.5.1 Informations à afficher :

- **pour les fonctions :** on affiche fonction puis le type de retour, le nom de la fonction et le numéro d'imbrication
- **pour les structures :** on affiche structure puis le type de structure et le numéro d'imbrication
- **pour les paramètres et attributs :** dans la table, nous avons d'abord l'identifiant, puis la caractéristique (paramètre ou attribut), le type et le déplacement

1.5.2 Implémentation :

- les objets `Tds` contiennent pour :
  - les fonctions `TdsFunction` le nom, le type, les paramètres et les attributs
  - les structures `TdsStruct` le nom et les attributs
  - les structures `TdsBloc` les attributs

1.5.3 Exemple de table des symboles :

Voici le résultat obtenu pour la table des symboles (figure 1). D'autres exemples sont présents sur le dépôt voici les liens
pour les tests :
`https://gitlab.telecomnancy.univ-lorraine.fr/Sami.Boulechfar/khatib3u/-/tree/master/Code/examples/Test_Semantique`
et les tables des symboles :
`https://gitlab.telecomnancy.univ-lorraine.fr/Sami.Boulechfar/khatib3u/-/tree/master/Code/out/tds/svg/Test_Semantique`

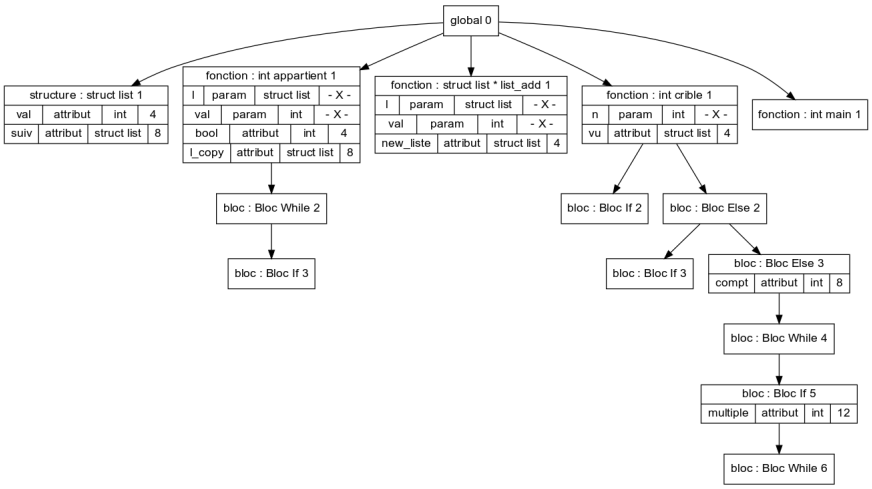

FIGURE 1 – Exemple de table des symboles pour crible.exp

1.6 Contrôles sémantiques :

La TDS étant un outil puissant pour l'analyse sémantique, l'idée était d'implémanter une liste de contrôles sémantiques qu'on a établie au sein de `TestSemantique` et de l'éxécuter au sein de `TdsVisitor`. Pour réaliser les tests, l'objet `TestSemantique` contient la liste des fonctions définies, des structures définies et la pile des blocs parcourus.

Table 1 – Liste des tests sémantiques

Test Sémantique	Description
main_non_def	main non ou mal défini (vérifie la signature)
Var_non_def	variable non définie (attention aux paramètres et attributs)
Var_deja_def	variable déjà utilisée (attention aux paramètres et attributs)
fonc_non_def	fonction non définie
fonc_deja_def	fonction déjà utilisée
struct_non_def	structure non définie
struct_deja_def	structure déjà créée
return_type	vérifie ce qui est renvoyé par la fonction
return_bloc	vérifie la présence du return. (attention si return dans bloc if)
type_param	type des paramètres d'une fonction
nombre_param	nombre paramètres correspond a ceux de la fonction
cond	vérifie qu'une condition est un booléen (pas x=1)
test_type	test les types entre les opérations
test_fleche_def	vérifie que la structure contient l'attribut
division par 0	ne pas écrire x/0
test_affectation	affectation possible (pas 3=3 ou addition()=3)

2 Les difficultés rencontrées :

2.1 Réalisation de la TDS :

La difficulté majeure du projet compil était la réalisation de la table des symboles. Ayant été établie lors des CMs et TDs en trad, la liste des informations contenues dans la TDS était assez claire. Cependant, la structure de données pour implémanter la TDS a été assez difficile à choisir. La première version de TDS réalisée a été implémantée grâce aux `ArrayList<>` et `HashMap<>`. L'idée était d'avoir une classe abstraite `TDSEntry` dont le comportement diffèrerait selon l'entrée. Ainsi des classes `TDSBlocEntry`, `TDSFuncEntry`, `TDSIfEntry`, `TDSParamEntry`, `TDSStructEntry`, `TDSVarEntry`, et `TDSWhileEntry` ont été implémantées, illustrant ainsi le comportement de la classe `TDSEntry` face à un bloc, une fonction, un if, une fonction à paramètres, une variable de type structure, une variable simple ou encore une boucle while. La limite de cette méthode résidait dans l'affichage de la TDS qui n'était pas assez intuitif ainsi que l'absence du stockage de l'information permettant la mise en place des contrôles sémantiques.

2.2 Réalisation des launchW :

Lors de la réalisation des launchW, la commande dot, issus de GraphViz n'est pas reconnu car issu de Linux. Cette commande permet de convertir un fichier .dot en fichier .svg. N'ayant pas trouvé d'équivalent sur Powershell, les tests sur Windows génèrent bien les fichiers .dot, mais il faut ensuite passer sur Linux pour convertir les .dot en .svg. Pour cela, un nouveau script sur linux a été écrit. Le fichier launch_dot_to_svg.sh permet de faire cette commande, que ça soit pour un ast, une tds, tous les ast ou toutes les tds.

3 Gestion de projet :

3.1 Diagramme de Gantt :

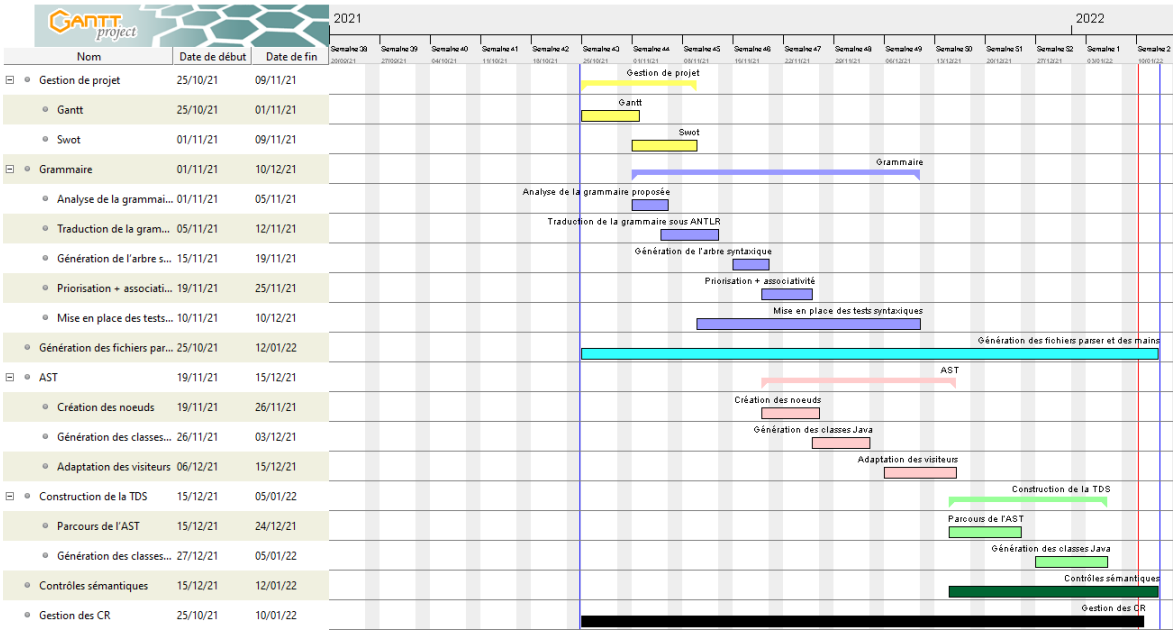

FIGURE 2 – Diagramme de Gantt

3.2 Tâches réalisées :

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

- Ecriture des tests : Hamza Abdoulhoussen et Sami Boulechfar
- Ecriture des launchs : Hamza Abdoulhoussen et Sami Boulechfar
- Ecriture de la tds : Maha Khatib, Hamza Abdoulhoussen et Sami Boulechfar
- Ecriture des tests sémantiques : Maha Khatib, Hamza Abdoulhoussen et Sami Boulechfar
- Ecriture du compte-rendu : Maha Khatib et Sami Boulechfar

3.3 Réunions de travail :

L'équipe a réalisé des réunions les 4 et 5 janvier pour réfléchir à la structure utilisée pour afficher la table des symboles et effectuer des contrôles sémantiques. L'équipe s'est ensuite répartie les contrôles sémantiques et les ont implémentés le week-end du 8-9 janvier. Les détails des réunions sont données en annexes.

3.4 Bilan personnel

3.4.1 Maha Khatib

Points positifs	- Investissement des membres du groupe et leur réactivité - Travail en présentiel des membres sur le projet régulièrement
Difficultés rencontrées	- Incertitudes concernant l'implémentation de la TDS
Expérience personnelle	- Application de la partie Front-End d'un compilateur en détail
Axes d'amélioration	- Gestion de la mise en place des tests au début du projet - Passer plus de temps sur les contrôles sémantiques

3.4.2 Hamza Abdoulhoussen

Points positifs	- Réalisation de launch permettant d'éxécuter facilement les commandes et les tests - Bonne répartition du travail
Difficultés rencontrées	- Difficultés à expliquer clairement le code réalisé et les structures utilisées
Expérience personnelle	- Utilisation du langage dot pour faire des schémas - Meilleure compréhension des premières étapes réalisées par un compilateur - Application du Visitor pattern
Axes d'amélioration	- Utilisation de merge request - Passer plus de temps à exposer les résultats lors des réunions

3.4.3 Sami Boulechfar

Points positifs	- Travail régulier et sérieux des membres - Progression en Java et dans l'utilisation du module GraphVizitor - Découverte du bash script
Difficultés rencontrées	- Absence d'entraînement pour l'implémentation de la TDS et des contrôles sémantiques - Difficulté à travailler ensemble durant les vacances de noël
Expérience personnelle	- Sujet par moment flou, par exemple la grammaire proposé était ambiguë - Projet portant parfois sur des notions jamais vu en TP.
Axes d'amélioration	- Mieux répartir les réunions le long du projet

3.5 Travail réalisé :

Etapes	Hamza	Sami	Maha
Gestion de Projet	3	4	8
Code - bash	10	8	4
Code - AST	10	8	4
Code - TDS	12	8	12
Code - contrôle sémantique	14	12	6
Test - Ecriture jeux d'essais	4	10	6
Rapports	3	8	8
Total	56h	58h	48h

4 Annexe :

4.1 Shéma récapitulatif de la PCL1 :

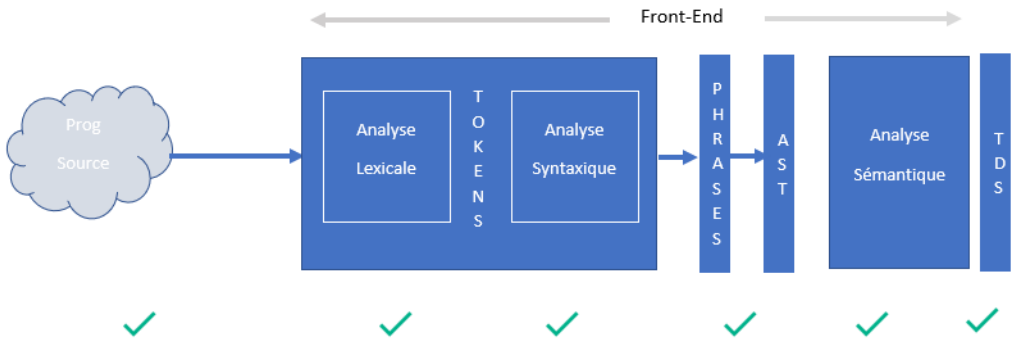

FIGURE 3 – Shéma compil

4.1.1 Mercredi 4 janvier 2022 :

Présent	Heure	Lieu
Maha KHATIB Sami BOULECHFAR Hamza ABDOULHOUSSEN	11h	TELECOM Nancy

Travail réalisé
L'équipe à d'abord fait le point sur le travail réalisé durant les vacances. Les membres ont ensuite réétudié la TDS pour savoir comment l'améliorer.

Objectifs suivants

- Avancer l'implémentation de la TDS
- Ajouter un launch pour tester la tds.

4.1.2 Jeudi 5 janvier 2022 :

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

Travail réalisé
La TDS étant terminée, l'équipe a ajouté de nouveau launch pour tester toutes les tds, tester les erreur syntaxiques et les erreurs semantiques. Elle a également converti les launchs sur Linux en launch sur Windows. Enfin, les contrôles sémantiques ont été listé sur feuille afin de mieux se répartir leurs implémentations.

Objectifs suivants

- Finir la rédaction du rapport d'activité
- Implémenter plus de tests sémantiques
- Améliorer le launchW

4.1.3 Vendredi 7 janvier 2022 :

Présent	Heure	Lieu
Maha KHATIB Sami BOULECHFAR Hamza ABDOULHOUSSEN	16h	TELECOM Nancy

Travail réalisé
Ajout de jeux d'essais pour tester les erreurs semantiques puis début d'implémentation des contrôles sémantiques.

Objectifs suivants

- Finir l'implémentation des jeux d'essais
- Améliorer le launchW
- Perfectionner le code