ENSEEIHT - SCIENCES DU NUMÉRIQUES

Rapport Projet TDL

Baptiste Gomez
Hermas Obou

Janvier 2025

Contents

1 Introduction

Dans ce projet, nous allons partir du compilateur RAT fait en TP et étendre ce langage aux pointeurs, variables globales, variables statiques locales et aux paramètres par défaut. Nous verrons dans ce rapport les choix de conceptions fait, les modifications de la grammaire de RAT ainsi que les difficultés que nous avons rencontrées.

2 Evolution des AST

- Lors de l'ajout des pointeurs, nous avons dû faire des modifications de la grammaire de RAT, notamment en ajoutant un nouveau symbole non terminal qui correspond aux affectables. Un affectable est un élément auquel on peut affecter une valeur. Cela engendre un tout nouveau type dans les AST qui va comprendre le déréférencement. De plus, une expression adment trois nouvelles entrées : "New", "Null" et "Adresse" pour la gestion des pointeurs.
- Pour l'ajout des variables globales et statiques locales, nous avons ajouté une "DeclarationStatic" et une "DeclarationGlobale" pour pouvoir différentier ces variables des variables classiques.
- Un nouveau type defaut était nécessaire pour l'ajout des paramètres par défaut pour pouvoir les séparer du reste et garder une trace de ceux-ci.

3 Jugements de typage

Expression

- $\sigma \vdash E : \tau$

  $\overline{\sigma \vdash A : \tau}$

- $\sigma \vdash TYPE : \tau$

  $\overline{\sigma \vdash (new \quad TYPE) : Pointeur(\tau)}$

- $\sigma \vdash Null : Pointeur(Undefined)$

Déréférencement

- $$\frac{\sigma \vdash A : Pointeur(\tau)}{\sigma \vdash (*A) : \tau}$$

Type

- $$\frac{\sigma \vdash TYPE : \tau}{\sigma \vdash TYPE* : Pointeur(\tau)}$$

Affectation

- $$\frac{\sigma \vdash A : \tau \qquad \sigma \vdash E : \tau}{\sigma, \tau_r \vdash A = E : void, []}$$

4 Pointeurs

Un pointeur est maintenant un affectable qui peut être déréférencer. Nous avons créer une fonction analyser-affectable à chaque passe pour gérer ce nouveau type. Un affectable admet les mêmes restrictions aux niveaux des identifiants que dans le langage RAT simple. De plus un pointeur agit comme une variable classique avec une taille de 1 à la phase de placement en mémoire.

5 Variables Globales

Les variables globales, du fait de leur déclaration en haut du code source, ne nécessitent pas énormément de changement. En effet, les points importants sont seulement à la déclaration des variables (double déclaration) et à l'affectation (bonne modification de la variable globale). Le placement mémoire et la génération de code ne posent pas de difficultés particulières.

6 Variables Statiques Locales

Nous avons ajouter en plus de la déclaration, la déclaration statique pour différentier les deux. La détection d'une déclaration statique dans le main se fait au niveau de la passe de placement et génère une erreure. Lors de la passe de placement en mémoire, nous avons différencier les blocs des blocs dans une fonction pour pouvoir intégrer la gestion des variables statiques loocales.

7 Paramètres par Défaut

Les paramètres par défaut ont nécessité la création d'un nouveau type "defaut" qui est en option dans parser.mly pour les garder en mémoire et faire le remplacement si nécessaire lors de AppelFonction dans la passe de gestion d'identifiant. Les passes de gestion d'identifiant, de typage et de génération du code TAM s'effectuent de la manière que pour une déclaration classique.

8 Conclusion

Les principales difficultés que nous avons rencontrés sont au niveau de Menhir et du parser.mly. Nous avons bloqué pendant longtemps sur des erreurs de syntaxe au niveau de Menhir lors de l'exécution du compilateur.De plus sur la gestion des parametre par de