

Rapport du Projet de Programmation Fonctionnelle et de Traduction des Langages

Etendre le compilateur du langage RAT

Kamal HAMMI
Mohamed IKICH

12 janvier 2022

1 Introduction

Ce document représente un petit rapport concis expliquant le travail réalisé lors du projet des matières Programmation Fonctionnelle et de Traduction des Langages. L'objectif était d'ajouter au compilateur du langage RAT réalisé en TP le traitement des nouvelles constructions : les pointeurs, l'opérateur d'assignation d'addition, les enregistrements et les types nommés.

2 Pointeurs

2.1 Evolution de la structure des AST

On remplace le type Ident de Expression par Affectable qui factorise l'identifiant et le déréférencement d'un affectable.

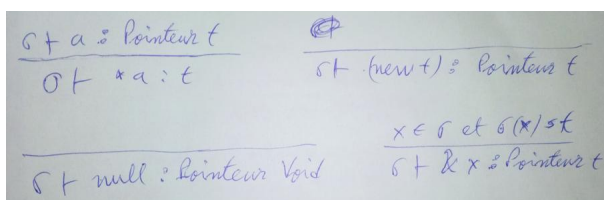
On ajoute le type NEW of typ qui est l'initialisation d'un Pointeur sur un type typ.

On ajoute Adresse d'un identifiant.

On ajoute la valeur NULL d'un Pointeur.

On change le type Affectation de l'instruction en ajoutant Affectable.

2.2 Jugements de Typages



$$\begin{array}{c}
 \frac{\sigma \vdash a : \text{Pointeur } t}{\sigma \vdash *a : t} \quad \frac{\sigma \vdash (new\ t) : \text{Pointeur } t}{\sigma \vdash \&x : \text{Pointeur } t} \\
 \frac{}{\sigma \vdash \text{null} : \text{Pointeur Void}} \quad \frac{x \in \sigma \text{ et } \sigma(x) \neq \text{null}}{\sigma \vdash \&x : \text{Pointeur } t}
 \end{array}$$

FIGURE 1 – Jugements de typage des Pointeurs

3 Assignation d'Addition

3.1 Evolution de la structure des AST

On ajoute le type Assignation d'addition dans l'instruction.

3.2 Jugements de Typages

$$\frac{\sigma \vdash a : \text{int} \quad \sigma \vdash E : \text{int}}{\sigma \vdash a + E : \text{void}, [\text{a}, \text{int}]}$$
$$\frac{\sigma \vdash a : \text{rat} \quad \sigma \vdash E : \text{rat}}{\sigma \vdash a + E : \text{void}, [\text{a}, \text{rat}]}$$

FIGURE 2 – Jugements de typage de l'assignation d'addition

4 Types Nommés

4.1 Evolution de la structure des AST

On ajoute le type `DefinitionType` dans l'instruction.
On change la structure du programme qui peut être un bloc ou un triplet d'une liste des couples (`nomType`, `typeCorrespond`) résultante des définitions des Types Nommés, puis une fonction après un sous programme.

4.2 Jugements de Typages

$$\frac{\sigma \vdash t : \text{Type Nommé } t'}{\sigma \vdash \text{typedef } n = t : \text{void}, [n, t']}$$
$$\frac{}{\sigma \vdash \text{typedef } n = t : \text{void}, [n, t']}$$

FIGURE 3 – Jugements de typage des Types Nommés

5 Enregistrements

5.1 Evolution de la structure des AST

On ajoute le type `Champ` dans `Affectable`.
On ajoute le type `Enregistrement` dans `Expression`.

5.2 Jugements de Typages

6 Explications sur Implementation

Pour Traiter les types récurssifs, on a utilisé une liste `list_types_recurssifs` qui stocke les types nommés et leur type `Struct` correspondant.

$$\begin{array}{c}
 \sigma \vdash (E_1 \dots E_n) : E_1 \\
 \hline
 \sigma \vdash \text{Enregistrement}(E_1 \dots E_n) : \text{TypeStruct } E_1 \\
 \\
 \sigma \vdash E_1 : E_1, \dots \sigma \vdash E_n : E_n \\
 \hline
 \sigma \vdash E_1 \dots E_n : E_1 \times \dots \times E_n
 \end{array}$$

FIGURE 4 – Jugements de typage des Enregistrements