Rapport de projet TDL

I Introduction

Pour ce projet, nous devions étendre le compilateur de RAT pour qu'il soit capable de traiter les **pointeurs**, **l'opérateur d'assignation d'addition**, la création de **types nommés** ainsi que les **enregistrements**. Il faudra donc adapter les analyseurs **lexicaux** et **syntaxiques** puis coder plusieurs passes en Ocaml, à savoir la passe de **gestion d'ID**, la passe de **typage**, la passe de **placement mémoire** et enfin la passe de **génération de code**.

II Pointeurs

II.A Type

Pour l'ajout des pointeurs dans un compilateur, on rajoute des TOKENS pour les mots clés suivants : **&**, **new**, **null**. Ensuite, on rajoute les **elements de la grammaire** définis dans le sujet relatifs au **pointeurs**. Ces changements permettent de modifier les différentes AST.
Pour l'AST syntaxe : on rajoute le type **Pointeur of typ** ainsi que le cas non terminal **Affectable** qui peut être un **Déréférencement d'Affectable** ou bien un **Indentifiant de String**. On modifie aussi le cas non terminal **Instruction** en y modifiant **l'affectation d'un Identifiant** par **l'affectation d'un affectable** aussi, le cas non terminal **Expression** sera modifié pour y ajouter les cas suivants : **Affectable d'Affectable**, **Null**, **Adresse de String**, **New de Typ**.
Pour l'AST TDS : la branche **Adresse de String** du type **Expression** devient **Adresse de Info_ast** et la branche **Identifiant de String** du type **Affectable** devient **Identifiant de Info_ast**.
Pour l'AST Type : Aucune modification nécéssaire.
Pour l'AST Placement : On ajoute un **entier** dans la branche **Affectable d'Affectable** du type expression pour la taille.

II.B Jugement de Typage

$$\frac{}{\sigma \vdash null : Pointeur(Undefined)}$$

$$\frac{\sigma \vdash T : \tau}{\sigma \vdash new(T) : Pointeur(\tau)}$$

$$\frac{\sigma \vdash Id : \tau}{\sigma \vdash \&Id : Pointeur(\tau)}$$

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

II.C modification des passes

Passe de gestion d'ID : Création d'une nouvelle fonction **analyse_tds_affectable tds modif a** qui prend en paramètre la **Tds**, l' **Affectable** ainsi que le booléen **modif** qui apporte l'information de si l' **Affectable** se trouve a droite ou a gauche d'un signe égal. Cette fonction se décompose de la manière suivante :
si l' **Affectable** est un **Indentifiant**, alors on **Recherche Globalement**, si on ne trouve pas, alors on lève une exception, si il existe, on traite en fonction du type.
Func→ exception
Const → si **Modif** alors **exception** sinon **EntierVal**
Var→ AstTds.Indent(Info).
si l' **Affectable** est un **Dereferencement**
→ **AstTds.deref(analyse_tds_affectable tds modif a)**.
Modification de la fonction **analyse_type_instruction** pour modifier le cas
Affectation(a,e) → let na = **analyse_tds_affectable tds true a** in
let ne = **analyse_tds_expression tds e** in **Affectation(na, ne)**
Ainsi que la fonction **analyse_tds_expression** pour ajouter les cas
Affectable(a) → let na = **analyse_tds_affectable tds faslse a** in **Affectable(na)**.
AstSyntax.Adresse(n) → recherche_globale
| **None** → **exception**
| **Some** → **Fun or Cst** → **exception**
| | → **var** → **AstTds.Adresse(Info)**
AstSyntax.Null → **AstTds.null**
AstSyntax.new(t) → **AstTds.New(t)**
On a aussi supprimé **Ident de String** du type **Expression**.

III Opérateur d'assignation d'addition

III.A Type

Pour l'ajout de**l'opérateur d'assignation d'addition** dans un compilateur, on rajoute des TOKENS pour les mots clés suivants : **PLUSEQ**. Ensuite, on rajoute les **elements de la grammaire** définis dans le sujet relatifs à **l'opérateur d'assignation d'addition**. Ces changements permettent de modifier les différentes AST.
Pour l'AST syntaxe : Dans instruction, on rajoute un type **Ajout of String**
Pour l'AST TDS : Dans instruction, on modifie le type **Ajout of String** en **Ajout of Info_Ast**
Pour l'AST Type : Dans instruction, on sépare le type **Ajout of String** en deux types
AjoutInt of tds.Info_Ast*expression et **AjoutRat of tds.Info_Ast*expression**

III.B Jugement de Typage

Aucune modification nécéssaire.

III.C modification des passes

Passe TdsRat : Pour la passeTdsRat, il a fallu modifier la fonction **Analyse_instruction**. En effet, lors du **match** de **ChercherGlobalement**, si on trouve quelque chose, on doit rajouter un match de la manière suivante :
match info_to_ast_info info **with**
| **InfoVar** _ →
let ne = **analyse_tds_expression tds e** in **Ajout(info, e)**
| _ → **exception**

Passe TypeRat : Pour la passeTypeRat, on a modifié la fonction **Analyse_type_instruction** pour ajouter un cas dans le **match** :

```
| AstTds.Ajout(ia, e) -> let t =(get_type ia) in
|     let(ne,te) = (analyse_type_expression e) in
|     |     if est_compatible t te then
|     |       if (analyse_type_binaire Plus t te = PlusInt) then
|     |         AstType.AjoutInt(ia, ne)
|     |       else AstType.AjoutRat(ia, ne)
|     |     else raise exception
```

Passe CodeRatToTam : Concernant la passeRatToTam, on ajoute deux cas dans le match, qui correspondent aux deux types de Ajout, en fonction du type d'expression sur laquelle on l'applique.

```
| AstType.AjoutRat(ia,e) →
|     let InfoVar(_,_,add,reg) = info_ast_to_info ia  in
|     |     let codee = analyse_expression(e,Rat) in (...)
| AstType.AjoutInt(ia,e) →
|     let InfoVar(_,_,add,reg) = info_ast_to_info ia  in
|     |     let codee = analyse_expression(e,Int) in (...)
```

IV Types Nommés

IV.A Type

non traité

IV.B Jugement de Typage

non traité

IV.C modification des passes

non traité

V Enregistrements

V.A Type

Pour l'ajout des enregistrements dans un compilateur, on rajoute des TOKENS pour les mots clés suivants : **.**, **Struct**.Ensuite, on rajoute les **elements de la grammaire** définis dans le sujet relatifs aux **Enregistrements**. Ces changements permettent de modifier les différentes AST.
Pour l'AST syntaxe :
On ajoute le type **Acces** = **Affectable*String** dans **Affectable** ainsi que le type
Creation = **Expression*List** dans **Expression**
Pour l'AST TDS : **Acces** devient **Affectable*info_Ast** dans **Affectable**
Pour l'AST Placement : Aucune modification.

V.B Jugement de Typage

$$\frac{\sigma \quad \vdash \quad Le : \tau}{\sigma \vdash Creation(Le) : Enre(\tau)}$$

V.C modification des passes

passe de gestion d'id :
Ajout d'un cas dans le match de **analyse_Affectable tds modif a**
| **Acces(Aff,nom)** = let na = **analyse_Affectable tds modif aff** in
| **Chercher_Globablement** →
| | **None** → **exception**
| | **Some** → **InfoFun or InfoConst** → **exception**
| | | → **InfoVar** → **InfoToInfoAst**
Ajout d'un cas dans le match de **analyse_expression tds a**
| **Creation(le)** → let **nle** = **(List.Map analyse_expression tds le)** in **AstTds.Creation(nle)**

VIII Conclusion

Durant ce projet, nous avons réalisé toutes les passes. Cependant, nous n'avons pas eu le temps d'implanter les types nommés. Le code ne presente aucun warning donc chaque filtrage a été completement défini. concernant les difficultés rencontrées, le test **testfun5** nous a fait perdre énormément de temps car il affichait 0 alors que la fonction ne renvoie rien.