TD4
TdL

Traduction des langages

Placement mémoire

Objectif :
- Définir la nouvelle structure d'arbre obtenue après la passe de placement mémoire
- Définir les actions à réaliser par la passe de placement mémoire

1 Placement mémoire du langage Rat pour une machine à pile

▷ **Exercice 1** *Donner le placement mémoire des variables du programme suivant :*

```
rat f3 (int a int b rat r){
  return [(a + num r) / (b + denom r)];
}

rat f2 (bool b rat x rat y){
  int x1 = num x;
  int x2 = denom x;
  rat res = call f3(x1 x2 y);
  return res;
}

int f1 (int i rat r int n){
  rat r1 = call f2(true r [i/n]);
  return denom r1;
}

test{
  int x = call f1 (13 [4/11] 17);
  rat y = call f2 (true [4/11] [11/4]);
  rat z = call f3 (1 2 [4/11]);
  print x;
  print y;
  print z;
}
```

2 Passe de placement mémoire

Nous rappelons qu'un compilateur fonctionne par passes, chacune d'elle réalisant un traitement particulier (gestion des identifiants, typage, placement mémoire, génération de code,...). Chaque passe parcourt, et potentiellement modifie, l'AST.

La troisième passe est une passe de placement mémoire.

2.1 Structure de l'AST post passe de placement mémoire

La passe de placement calcule l'adresse des variables dans la pile. Cette adresse doit être mise à jour dans les informations associées aux identificateurs.

▷ **Exercice 2** *Définir la structure de l'AST post passe de placement mémoire.*

2.2 Actions à réaliser lors de la passe de résolution des identifiants

▷ **Exercice 3** *Définir les actions à réaliser lors de la passe de placement mémoire.*

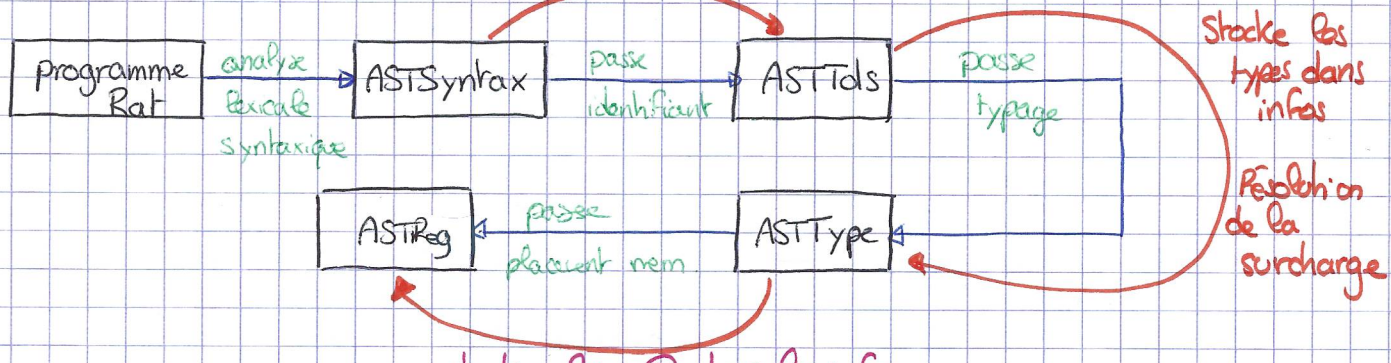
TD4
TdL

Placement mémoire

Exercice 1 : Exemple pour comprendre

f3	f2	f1	test
a -4 [LB]	b -5 [LB]	i -4	x 0 [SB]
b -3 [LB]	x -4 [LB]	r -3 [LB]	y 1 [SB]
r -2 [LB]	y -2 [LB]	n -1 [LB]	z 3 [SB]
	x1 3 [LB]	r1 3 [LB]	
	x2 4 [LB]		
	res 5 [LB]		

Exercice 2 : Structure AST Remplace les id par leur info

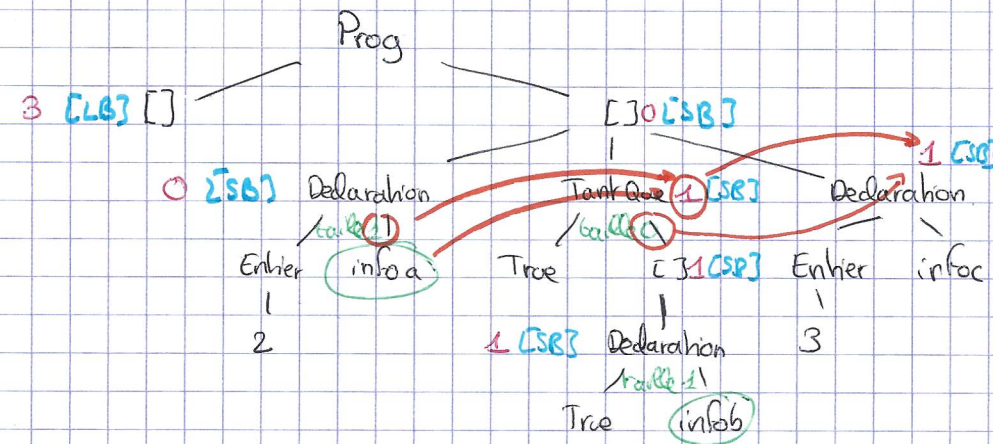


- stocker les @ dans les infos
- possibilité de supprimer la liste des paramètres dans la def d'une fonction.

Exercice 3 : Actions à réaliser

- registre en paramètre des fonctions
- rep en paramètre des fonctions
- besoin d'une fonction de taille des instructions.

```
main {
  int a = 2;
  while (true) {
    bool b = true;
  }
  int c = 3;
}
```



```
let getTaille i =
    match i with
      | Declaration (e, info) -> getTaille (getType info).
      | _ -> 0

let analyser AstType.Prog (fonctions, bloc) =
    let nF = List.map (analyse_dep_fonction "LB") in
             analyse_bloc 0 "SB" bloc;
    Prog (nf, bloc).

let analyse_dep_bloc dep reg li =
(* int -> string -> ASTType.instruction list -> unit *)
    match li with
      | [] -> ()
      | i :: qi -> analyse_dep_instruction dep reg i ;
                   analyse_dep_bloc (getTaille i + dep) reg qi;

let analyse_instruction dep reg i =
    match i with
      | Declaration (e, info) -> mise_a_jour_adresse info dep reg
      | Conditionnelle (e, b1, b2) -> analyse_dep_bloc dep reg b1;
                                      analyse_dep_bloc dep reg b2
      | TantQue (e, b) -> analyse_dep_bloc dep reg b
      | _ -> ()
```