Chapitre 4 – Arbres abstraits (AST) / Table des symboles (TDS) / Contrôles sémantiques

Génération de code + optimisation

Analyse sémantique

Analyse   
Lex | Synt

Texte Source

Génération des noms

Arbre abstrait  
(± TDS)

Arbre abstrait

(Décoré ou TDS)

Analyse Sémantique : Vérifie que le texte source(Arbre abstrait) est conforme à ensemble de règles de construction des programmes définis par le manuel de référence du langage.

Exemples :

* Variables déclarées avant leur utilisation
* Doubles déclarations interdites
* Compatibilité des types
* …
* Deux catégories d’erreurs sémantiques :
  + Erreurs sémantiques statiques : « Vérifiable » à la compilation
    - Cohérence de types
    - Variables correctement déclarées
      * Dépend des règles de
        + Portée des déclarations
        + Visibilité des variables
  + Erreurs sémantiques dynamiques : « Vérifiable » à l’exécution (A prévoir dans le code généré)
    - Division par 0 (X/Y)
    - Allocation mémoire
    - Dépassement tableau
* Un arbre abstrait : un arbre syntaxique dans lequel on ne conserve que ce qui nous servira pour la production de code et le contrôle sémantique.
* La portée d’une déclaration : La portion de code où elle est définie { … }. Les variables sont disponibles dans les sous-blocs si le manuel d’utilisation du langage le permet.
* La visibilité : d’une variable, la portion du programme où elle utilisée.
* Les règles de visibilité : détermine à quelles définitions d’un identificateur une utilisation est susceptible de se rapporter
* Un bloc :
  + Un identificateur
  + Procédure
  + Fichier (en c)
  + { … } : Un bloc sans nom
* En général, les déclarations sont uniques dans un bloc
  + Exemple : int y ; char y;
* Espace des noms (précisé dans le manuel de référence du langage) : ensemble de ce qui est désorganisée dans un contexte donne par 1 méthode d’accès faisant usage de noms symbolique
  + Exemple en C :
    - Struct un\_entier{  
       int i ;  
      } i ;  
      On peut écrire i.i = 3 ; => Correcte car l’espace des noms est différent.
    - En C, il y a plusieurs espaces des noms :
      * Étiquette
      * Enum, struct, unions
      * Les variables, les fonctions, les types
      * Les champs de structures, unions

A retenir :

* Une variable est unique dans son espace des noms.
* Un même identificateur peut-être déclaré n fois dans un bloc, uniquement s’il appartient à n espaces des noms différents

AST

Init

Décl

* Identificateur : pour toutes utilisation d’un identificateur, trouver sa déclaration correspondante
* Les mécanismes d’identification précèdent l’étape de contrôle du typage.
* Comment représenter cette correspondance ?
  + Utilisation d’un identificateur et sa déclaration.
* 3 représentations :
  + À la sortie de l’arbre abstrait (AST) : un AST complet

Un pointeur vers le nœud de la déclaration

Arbre abstrait décoré



Décl

X = 3

* + AST des instructions + TDS  
    Début de TDS : Table des symboles (1 par bloc)  
    TDS : Construite au cours de l’analyse syntaxique  
      
      
     TDS

…

Nature

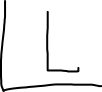
Type entier

« X »

Init

X = 3

g()  
 f(z : entier)  
 x entier ;  
 y entier ;  
 x := y + 3 ;  
 x := 3



* Première idée des infos à mettre dans la TDS :
  + Le nom de la variables « x »
  + Son type
  + Sa nature
    - var, fonction -> nb param, type, param -> mode de passage), pour les tableaux : (dimension, taille via les barres)
    - Déplacements

Chaque entrée est une variable déclarée du bloc.

Question à l’examen qui sont souvent sur TDs  
  
  
Une fonction f : DEBUT

F() 🡪 Définition de f

{ MOVL

X : entier ADD

Y : réel BENZ

SSR

Z : entier 🡺Main()

X = Z + Y {

} x : entier

F()

X : 10 ;

SP

}

BASE

Déplacement

|  |  |
| --- | --- |
| … | … |
| Z | 0 |
| Y | 0 |
| X | 0 |
| { | … |
| X | 0 |
|  |  |

A quoi sert le déplacement dans la TDS ? Pour pouvoir accéder à la variable dans la mémoire à l’exécution  
  
code \_f. {

Réserver 8 octets

À l’éxécution

Inst de f

{ADD #3, @3

TDS

BASE + DÉPALCMEENT

X a déplacement de 0  
Y a déplacement de 2  
Z a déplacement de 6

Décl

X = 3

TDS f()

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

CONSTRUIT UNE TDS par bloc

La partie « Déclaration » de l’AST devient inutile pour la génération de code.

Comment réaliser l’étape d’identification ?

On associe à un identificateur sont unique déclaration.

On a un AST

des TDS

F1

X : entier

F2

Y : réel

F3

F4

Z = 10

X = 5

F5

Z : entier

Z = 5

TDS F1

X : entier

TDS F2 :

Y : réel

TDS F3 :

----

TDS F4 :

----

TDS F5 :

Z : char

|  |
| --- |
|  |
|  |
| … |
| 4 |
| 3 |
| 2 |
| 1 |

Pile des régions : La pile des régions sert à trouver l’unique déclaration pour une entrée donnée.

A quoi ça sert ?

Pile des régions : La pile des régions sert à trouver l’unique déclaration pour une entrée donnée.