Construction de la table des symboles

Alexis Nasr
Carlos Ramisch
Manon Scholivet
Franck Dary

Compilation – L3 Informatique
Département Informatique et Interactions
Aix Marseille Université

Principe

- La construction de la table des symboles est réalisée lors d'un parcours de l'arbre abstrait.
- Ce parcours est réalisé à l'aide d'une classe visiteur, qui étend `SaDepthFirstVisitor`.
- Les classes implémentant la la table des symboles se trouvent dans le package `ts`.
- A un programme correspond :
  - Une table globale, qui possède une entrée pour toute variable globale et toute fonction
  - Autant de tables locales que le programme comporte de fonctions.

Le package `ts`

- Une seule classe `Ts` est utilisée pour la table globale et les tables locales.
- Elle est constituée de :
  - Deux tables de hashs `variables` et `fonctions`, qui associent à un identificateur de variable ou de fonction l'entrée lui correspondant. Dans le premier cas, l'entrée est une instance de la classe `TsItemVar` et dans le second, une instance de la classe `TsItemFct`.
  - Deux entiers `adrVarCourante` et `adrArgCourant` qui indiquent la taille occupée par les variables locales et les paramètres de la fonction (pour les tables locales).
- Les entrées des tables des symboles sont des instances d'une des deux classes suivantes :
  - `TsItemVar` Entrée correspondant à une variable ou à un paramètre
  - `TsItemFct` Entrée correspondant à une fonction (uniquement pour la table globale)

Classes du package `ts`

- `TsItemFct` définit les variables d'instance suivantes :
  - `String identif` : le nom de la fonction.
  - `int nbArgs` : son nombre d'arguments.
  - `Ts table` : table des symboles locale à la fonction.
  - `SaDecFonc saDecFonc` : le nœud de l'arbre abstrait correspondant à la déclaration de la fonction.
- `TsItemVar` définit les variables d'instance suivantes :
  - `String identif` : le nom de la variable.
  - `int taille` : la taille mémoire occupée par la variable. Elle est égale à un pour des variables simples et, pour les tableaux, à la taille de ces derniers.
  - `Ts portee` : la portée de la variable, représentée par la table à laquelle appartient la variable.
  - `boolean isParam` : indique s'il s'agit d'une variable ou d'un paramètre.
  - `int adresse` : adresse relative de la variable. La première variable introduite dans la table a pour adresse 0, la seconde 0 + la taille de la première variable...

Liens entre les tables des symboles et l'arbre abstrait

`Sa` $\rightarrow$ `Ts`

- `SaDecFonc` définit : `tsItem`
  dont la valeur est le `TsItemFct` de la fonction définie.
- `SaDecVar` et `SaDecTab` définissent : `tsItem`
  dont la valeur est le `TsItemVar` de la variable déclarée.
- `SaAppel` définit : `tsItem`
  dont la valeur est le `TsItemFct` de la fonction appelée.
- `SaVarSimple` et `SaVarIndicee` définissent : `tsItem`
  dont la valeur est le `TsItemVar` de la variable référencée.

`Ts` $\rightarrow$ `Sa`

- `TsItemFct` définit : `saDecFonc`
  dont la valeur est le nœud de l'arbre abstrait correspondant à la définition de la fonction.

Parcours de l'arbre abstrait

- Tout nœud de l'arbre abstrait définit la méthode

```
public <T> T accept(SaVisitor <T> visitor) {
    return visitor.visit(this);
}
```

- La classe `SaDepthFirstVisitor` définit pour chaque type de nœud `X` de l'arbre abstrait la méthode

```
public T visit(X node)
{
    defaultIn(node);
    ...
    defaultOut(node);
    return null;
}
```

Exemple

```
public T visit(SaInstSi node)
{
    defaultIn(node);
    node.getTest().accept(this);
    node.getAlors().accept(this);
    if(node.getSinon() != null) node.getSinon().accept(this)
    defaultOut(node);
    return null;
}
```

Sa2ts

- `Sa2ts` étend la classe `SaDepthFirstAdapter`

```
public class Sa2ts extends SaDepthFirstVisitor <Void>
```

- et redéfinit les méthodes suivantes :
  - `Void visit(SaDecVar node)`
  - `Void visit(SaDecTab node)`
  - `Void visit(SaDecFonc node)`
  - `Void visit(SaVarSimple node)`
  - `Void visit(SaVarIndicee node)`
  - `Void visit(SaAppel node)`