Construction de l'arbre abstrait avec SableCC

Alexis Nasr
Carlos Ramisch
Manon Scholivet
Franck Dary

Compilation – L3 Informatique
Département Informatique et Interactions
Aix Marseille Université

Principe

- La construction de l'arbre abstrait se fait lors d'un parcours de l'arbre de dérivation à l'aide du visiteur `DepthFirstAdapter` généré par `Sablecc`.
- Une classe pour chaque type de nœud, définie dans le package `sa`.

Le package sa

Classe	Interface	implémente	hérite de
	SaNode		
	SaVar SaDec SaExp SaInst		SaNode SaNode SaNode SaNode
SaInst*		SaInst	
SaExp*		SaExp	
SaDecVar SaDecFonc SaDecTab		SaDec SaDec SaDec	
SaVarIndicee SaVarSimple		SaVar SaVar	
SaAppel		SaExp, SaInst	
SaLExp SaLInst SaLDec		SaNode SaNode SaNode	
SaProg		SaNode	

Parcours de l'arbre de dérivation

- Le visiteur `DepthFirstAdapter` permet de réaliser un parcours en profondeur de l'arbre de dérivation.
- Il définit pour chaque type de nœud `X` la méthode :

```
public void caseX(X node)
```

- Qui réalise le parcours d'un nœud de la classe `X`.
- Exemple pour la classe `APlusExp3` :

```
public void caseAPlusExp3(APlusExp3 node){
    inAPlusExp3(node);
    if(node.getExp3() != null) node.getExp3().apply(this);
    if(node.getPlus() != null) node.getPlus().apply(this);
    if(node.getExp4() != null) node.getExp4().apply(this);
    outAPlusExp3(node);
}
```

- Les classes correspondant aux nœuds de l'arbre de dérivation définissent la méthode

```
public void apply(Switch sw)
```

- Exemple de la classe `APlusExp3` :

```
public void apply(Switch sw)
  {((Analysis) sw).caseAPlusExp3(this);}
```

Parcours de l'arbre de dérivation

- On définit un visiteur `Sc2sa` fondé sur `DepthFirstAdapter`

```
public class Sc2sa extends DepthFirstAdapter
```

- Qui définit la méthode `caseX(X node)` pour chaque type de nœud `X`.
- Pour la classe `APlusExp3` on aurait aimé écrire :

```
// exp3 = {plus} exp3 plus exp4
public SaExp caseAPlusExp3(APlusExp3 node){
    SaExp op1 = node.getExp3().apply(this);
    SaExp op2 = node.getExp4().apply(this);
    return new SaExpAdd(op1, op2);
}
```

- Mais les méthodes `apply` et `case` ne renvoient rien (`void`).

```
public void apply(Switch sw)
public void caseAPlusExp3(APlusExp3 node)
```

Parcours de l'arbre de dérivation

- On définit dans la classe `Sc2sa` une variable d'instance `returnValue`, qui permet de simuler le retour des fonctions.
- On écrit `caseAPlusExp3` de la façon suivante :

```
public class Sc2sa extends DepthFirstAdapter{
    private SaNode returnValue;
    // exp3 = {plus} exp3 plus exp4
    public void caseAPlusExp3(APlusExp3 node)
    {
        SaExp op1 = null;
        SaExp op2 = null;
        node.getExp3().apply(this);
        op1 = (SaExp) this.returnValue;
        node.getExp4().apply(this);
        op2 = (SaExp) this.returnValue;
        this.returnValue = new SaExpAdd(op1, op2);
    }
```