La compilation

Le langage miniJava

UV IDL

1 Le typage

Vu la sémantique du langage miniJava, le typage doit se dérouler en deux phases :

1. La construction de l'environnement de définition des classes. Cet environnement contient le type des méthodes de chaque classe. Cette phase ignore les attributs (qui ne sont pas visibles en dehors de la classe) et les corps des méthodes.
2. La deuxième phase s'occupe de vérifier que *l'intérieur* des classes est correct (principalement le corps des méthodes). Elle va également s'assurer de la correction de l'expression de plus haut niveau.

Pour anticiper l'étape suivante (l'évaluation), je vous conseille d'étendre tous les nœuds de l'arbre syntaxique abstrait en ajoutant l'information de type.

2 L'exécution de MiniJava

L'exécution d'un programme MiniJava se fera par interprétation d'une forme pré compilée. Cette phase correspond à deux actions : (1) la compilation du programme et (2) son exécution.

2.1 La compilation

Le but de cette première partie est de construire la version exécutable du programme.

Chaque objet sera alloué lors de l'exécution sur le *tas*. Ce descripteur d'objet contiendra les valeurs de tous les attributs (variables d'instance) de l'objet ainsi qu'un lien vers le descripteur de sa classe.

2.1.1 Descripteur d'objet

L'arrangement des attributs dans le descripteur d'objet est réalisé selon une forme appelée préfixe. Le but est que pour tout objet du type T qui possède un certain attribut a, cet attribut figure toujours à la même position. Ceci n'est possible que parce que l'héritage est simple. Ainsi par exemple, si nous disposons de deux classes `A` et `B` définies par le code suivant :

```
class A { Int x; Boolean b; }
class B extends A { String str; }
```

Chaque objet `a` instance de `A` et `b` instance de `B` auront un descripteur de la forme :

descripteur de `a` :

x
b

descripteur de `b` :

x
b
str

Ainsi, lors de la compilation, il faudra construire la forme de ces descripteurs d'objet pour chaque classe.

2.1.2 Descripteur de classe

Les classes (ou plutôt les descripteurs de classes) sont des entités statiques allouées lors de la compilation. Dans les compilateurs, ses éléments sont souvent allouées dans l'espace des variables globales à savoir le fond de la pile.

Dans le cadre de notre projet, ces descripteurs sont alloués dans une zone des classes. Il mémorise la classe mère et les méthodes de la classe.

Dans le cadre d'une sémantique objet à la Java, la gestion des méthodes peut être simple et suivre le modèle préfixe des attributs. La méthode `m` d'une classe `X` est renommée en `X_m` qui est unique[1]. Ces méthodes sont alors stockées dans l'espace des méthodes (dans notre cas, une table des méthodes). Alors, le descripteur de classe aura la forme ci-dessous pour les deux classes suivantes :

```
class A { int f() {...} boolean g() {...} }
class B extends A { int f() {...} boolean h() {...} }
```

descripteur de `A` :

A_f
A_g

descripteur de `B` :

B_f
A_g
B_h

2.1.3 Cas particuliers

Trois objets particuliers représentent des constantes et sont alloués à la compilation `null`, `true` et `false`.

Certaines classes sont prédéfinies : Par exemple, `Object`, `String`, `Int` et `Boolean`. Leur gestion se fait également par des descripteurs de classes, la seule différence est l'implantation de leurs méthodes qui peut être fait en Ocaml et donc non obtenu par compilation.

1. Si la surcharge est autorisée, il faudra rendre plus complexe le schéma de renommage. On utilise alors les types des arguments pour construire systématiquement un nom unique.

2.1.4 Contenu de la phase de compilation

En résumé, la phase de compilation devra suivre les étapes suivantes :

1. Allocation des classes particulières
2. Allocation des objets particuliers
3. Allocation des classes : par une méthode `compile(C)` qui suit l'algorithme suivant. Si `C` n'est pas déjà compilée, compiler la classe mère de `C` puis :
   (a) ajouter les méthodes de `C` à la table des méthodes,
   (b) construire le descripteur de `C`,
   (c) construire la table pour les attributs de `C`,

2.1.5 Attributs et méthodes statiques

Si vous avancez bien, vous pouvez également vous attaquer à l'implantation des attributs et méthodes statiques (*i.e.* de classes). Les variables de classes sont des valeurs stockées dans la classe et peuvent donc être gérées dans le descripteur de classe sous la forme d'un descripteur d'objet...Les méthodes statiques sont elles résolubles à la compilation, ainsi tout appel à une méthode statique sera t'il remplacé par un accès direct à son corps.

2.2 L'exécution

L'exécution du programme consiste alors à évaluer l'expression principale en utilisant toutes les tables construites lors de la phase de compilation.