TP compilation : compilateurs CLS → EXT → REC

Le langage CLS est un langage à objets, compilé en deux étapes vers REC.

1 Langage EXT

Le langage EXT étend le langage REC avec la possibilité de déclarer qu'un nouveau type de structure étend un type de structure pré-existant. Ainsi on peut écrire

```
struct a { x; y }
struct b extends a { z }
```

pour définir deux types de structures `a` et `b`, les structures `a` ayant deux champs `x` et `y`, et les structures `b` ayant trois champs `x`, `y` et `z`.

Syntaxe En dehors de la possible inclusion du mot-clé `extends` lors de la déclaration d'un type de structure, la syntaxe concrète et la syntaxe abstraite de EXT sont identiques à celles de REC.

Sémantique de l'extension Lorsqu'une déclaration

```
struct a { x; y }
struct b extends a { z }
```

a été faite, on veut que la structure `b` soit un cas particulier de la structure `a`. En particulier, toute valeur de type `b` doit pouvoir être utilisée dans un contexte où l'on attendait une valeur de type `a`. L'accès à `*(v).x` par exemple, doit se faire de la même manière que `v` soit de type `a` ou de type `b`.

Si une structure fille déclare un champ de même nom que l'un des champs de la structure mère, alors les deux sont confondus.

Traduction vers REC La traduction de EXT vers REC consiste uniquement à traduire les déclarations de types de structures, de sorte que à ce que chacune soit complète.

Notez que pour respecter la sémantique de l'extension dans notre compilateur, il faut que les champs hérités apparaissent aux mêmes positions dans la structure fille et la structure mère. En particulier, si une structure fille déclare un champ de même nom que l'un des champs de la structure mère alors la position commune de ces deux champs doit être celle donnée par la structure mère.

2 Langage CLS

Le langage CLS permet de définir des classes, que l'on peut voir comme des structures regroupant des valeurs (les attributs) et des définitions de fonctions (les méthodes). Une classe fille peut hériter d'une classe mère. Une classe fille peut redéfinir certaines méthodes de sa classe mère. Une classe spéciale nommée `_master` est mère de toutes les classes qui ne sont pas elles-mêmes définies pas extension.

Syntaxe Les déclarations d'attributs se font comme les déclarations de champs de REC, et les définitions de méthodes se font comme les définitions de fonctions de FUN.

```
class c {
  var a;
  var b;
  init(va, vb) { a := va; b := vb; return(0); }
```

```
    f() { print(99); return(0); }
    g() { print(a+b); return(0); }
  }
```

Compilation vers EXT Chaque définition de classe `a` va produire deux définitions de structures :

- Un type de structure `a_descr`, appelé *descripteur de classe* de `a`. Un unique élément de ce type est instancié au début du programme. Il contient un pointeur vers le descripteur de la classe mère de `a` ainsi que les pointeurs vers les méthodes de `a`.
- Un type de structure `a`, représentant les instances de cette classe. Il contient un pointeur vers le descripteur de classe associé ainsi que les attributs.

Au début de l'exécution du programme, on initialise les descripteurs de toutes les classes utilisées.

Chaque définition de méthode à n paramètres est traduite par une fonction à $n+1$ paramètres. Le paramètre supplémentaire est appelé `this` et est passé en première position. Il correspond au paramètre implicite de la méthode, c'est-à-dire à l'objet à partir duquel la méthode est appelée.