UFR 919 Informatique – Master Informatique

Spécialité STL – UE ILP

TME8 — Accès réflexifs aux champs

Christian Queinnec

Ce TD/TME est basé sur l'examen de janvier 2006.

1 Présentation

Le but de ce problème est de faire travailler certains aspects réflexifs présents dans les bibliothèques (Java et C) d'exécution d'ILP6. Les différents dialectes d'ILP sont de la classe de Javascript qui autorise que l'accès au champ `x` de l'objet `o` puisse non seulement s'écrire `o.x` mais aussi `o["x"]` ce qui permet de calculer le nom du champ dont on souhaite obtenir la valeur.

Lorsqu'est calculable le nom du champ digne d'intérêt, trois actions sont possibles : déterminer si le champ existe, lire sa valeur, la modifier (ou la créer si elle n'existait pas déjà). En Javascript, ces opérations ont pour syntaxes :

```
"x" in o
o["x"]
o["x"] = expression
```

La première graphie permet de déterminer s'il existe un champ `x` dans l'objet `o`. Le résultat est booléen.

La graphie `o["x"]` peut s'utiliser à gauche ou à droite du signe d'affectation. Elle repose sur le fait qu'un objet est aussi une table associative indexée par des chaînes de caractères. Les champs de l'objet, tels que définis par la classe de l'objet, sont aussi accessibles par leur nom. Ainsi peut-on écrire :

```
class Point Object {
   field x;
   field y;
}
let p = new Point(11, 22)
in print(p.x + p["y"] + (new Point(10, 20))["x" + ""]))
```

Objectif : Àjouter à ILP6 ces accès dynamiques aux champs des objets.

Buts :

— Se familiariser avec ILP6
— Comprendre l'évaluation en ILP6
— Comprendre la compilation en ILP6

Note : Les questions de compilation d'ILP6 sont à effectuer après le cours associé.

2 Étapes

Pour vous aider, télécharger le fichier tme8.jar qui contient les interfaces dont vous aurez besoin, les fichiers `ilpField.*`, un nouveau fichier template et un nouveau script bash pour la compilation.

2.1 Grammaire

Écrire une grammaire RelaxNG pour `ilp6tme8`. Nommer ce fichier `grammar6tme8.rn[c|g]`. Les trois nouvelles expressions seront nommées, respectivement, `champExistence`, `champLecture` et `champEcriture`.

2.2 Existence

Écrire la classe `fr.upmc.ilp.ilp6tme8.CEASTchampExistence` et la méthode d'interprétation associée. On pourra sauter dans un premier temps, les méthodes annexes pour se concentrer sur `eval6`.

2.3 Compilation d'existence

Compléter la précédente classe `fr.upmc.ilp.ilp6tme8.CEASTchampExistence` avec une méthode `compile` implantant la compilation vers C. Attention en modifiant cette classe à ne pas détruire le comportement obtenu à la question précédente !

Pour ce faire, on écrira notamment une fonction en C, nommée `ILP_find_field`, prenant un objet ILP et le nom d'un champ (une chaîne C) et retournant un objet ILP de type `ILP_Field` représentant ce champ s'il existe. On utilisera cette fonction pour alléger le code des questions qui suivront.

Il est plus qu'utile de réfléchir à la forme générale du résultat de compilation avant de se lancer dans le détail du code C.

2.4 Lecture

Écrire la classe `fr.upmc.ilp.ilp6tme8.CEASTchampLecture` et la méthode d'interprétation associée.

La lecture d'un champ qui n'existe pas dans un objet conduit à une exception.

On pourra étendre la question en permettant d'écrire, en ILP6tme8, `o["print"]` ce qui rend possible l'obtention d'une méthode, par son nom.

2.5 Compilation de lecture

Compléter la précédente classe `fr.upmc.ilp.ilp6tme8.CEASTchampLecture` avec une méthode `compile` implantant la compilation vers C.

Pour ce faire, on écrira une fonction en C, nommée `ILP_find_field_value_address`, prenant un objet ILP et le nom d'un champ (une chaîne C) et retournant l'adresse de la zone mémoire où se trouve (si elle existe) l'adresse de la valeur de ce champ (le type C de retour est donc `ILP_Object*`). On utilisera cette fonction pour alléger le code des questions qui suivent.

2.6 Écriture

En supposant que le champ ciblé existe bien dans l'instance, écrire la classe `fr.upmc.ilp.ilp6-tme8.CEASTchampEcriture` et la méthode d'interprétation associée.

L'écriture d'un champ qui n'existe pas dans un objet conduit à la création de ce champ dans cet objet (dans cet objet seulement et pas dans sa classe). Comment implanteriez-vous cela ?