Examen final ILP
durée 3 heures
Revision: 1.7

Christian Queinnec

23 janvier 2006

Conditions générales

Cet examen est formé d'un unique problème en plusieurs questions auxquelles vous pouvez répondre dans l'ordre qui vous plait.

Le barème est fixé à 20 ; la durée de l'épreuve est de 3 heures. Tous les documents sont autorisés et notamment ceux du cours[1].

Votre copie sera formée de fichiers ASCII que vous rassemblerez dans un répertoire nommé `ilp` (sans aucun sous-répertoire) placé immédiatement dans votre répertoire personnel (ou `HOME`). Le répertoire `ilp` (pas votre `HOME`) sera ramassé automatiquement en fin d'examen par le centre de calcul. Seuls les fichiers mentionnés dans les livraisons à effectuer seront ramassés. Aucun fichier des sources d'ILP n'est à modifier.

Pour vous aider, un script vous est proposé. Vous le lancerez ainsi :

```
% source /Infos/lmd/2005/master/ue/ilp-2005oct/E/install.sh
```

L'examen sera corrigé à la main, il n'est donc pas utile de s'acharner sur un problème de compilation. Il est beaucoup plus important de rendre aisé, voire plaisant, le travail du correcteur.

Vous pouvez vous aider des machines pour naviguer dans la documentation ou dans le code d'ILP6 (avec Emacs (`etags`) ou Eclipse). Attention, il est peu conseillé que vous lanciez tous Eclipse en même temps : Eclipse n'est pas nécessaire pour se balader dans le code.

1 Objets dynamiques

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.

[1] `http://www.infop6.jussieu.fr/2005/IMG/queinnec/`

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"])
```

Ce nouveau dialecte, basé sur ILP6, sera nommé `ilp6dyn`. Le paquetage `fr.upmc.ilp.ilp6dyn` sera donc le paquetage que vous aurez à compléter.

Question 1

Écrire une grammaire RelaxNG pour `ilp6dyn`. Les trois nouvelles expressions seront nommées, respectivement, `champExistence`, `champLecture` et `champEcriture`.

Livraison

– un fichier *grammar6dyn.rnc*

Notation sur 1 point

– 1 point

Question 2

Écrire la classe `fr.upmc.ilp.ilp6dyn.CEASTchampExistence` et la méthode d'interprétation associée. On n'oubliera pas la force de persuasion que peut revêtir un croquis (en ASCII-art) bien pensé ou un commentaire pertinent.

Livraison

– un fichier Java nommé `CEASTchampExistence.java`

Notation sur 3 points

– 3 points

Question 3

Compléter la précédente classe `fr.upmc.ilp.ilp6dyn.CEASTchampExistence` avec une méthode `compile` implantant la compilation vers C. Attention en modifiant cette classe de ne pas détruire le comportement obtenu à la question précédente! On n'oubliera pas la force de persuasion que peut revêtir un croquis bien pensé ou un commentaire pertinent. Il sera notamment plus qu'utile d'indiquer la forme générale du résultat de compilation avant de se lancer dans le détail du code C.

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.

Livraison

– un fichier Java nommé `CEASTchampExistence.java`
– un fichier C nommé `ilpField.c`

Notation sur 4 points

– 4 points

Question 4

Écrire la classe `fr.upmc.ilp.ilp6dyn.CEASTchampLecture` et la méthode d'interprétation associée. On n'oubliera pas la force de persuasion que peut revêtir un croquis bien pensé ou un commentaire pertinent.

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

Livraison

– un fichier Java nommé `CEASTchampLecture.java`

Notation sur 3 points

– 3 points

Question 5

Compléter la précédente classe `fr.upmc.ilp.ilp6dyn.CEASTchampLecture` avec une méthode `compile` implantant la compilation vers C. Attention en modifiant cette classe de ne pas détruire le comportement obtenu à la question précédente! On n'oubliera pas la force de persuasion que peut revêtir un croquis bien pensé ou un commentaire pertinent. Il sera notamment plus qu'utile d'indiquer la forme générale du résultat de compilation avant de se lancer dans le détail du code 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.

Livraison

– un fichier Java nommé `CEASTchampLecture.java`
– un fichier C nommé `ilpField.c`

Notation sur 4 points

– 4 points

Question 6

Écrire la classe `fr.upmc.ilp.ilp6dyn.CEASTchampEcriture` et la méthode d'interprétation associée. On n'oubliera pas la force de persuasion que peut revêtir un croquis bien pensé ou un commentaire pertinent.

L'écriture d'un champ qui n'existe pas dans un objet conduit à la création de ce champ dans cet objet. N'oubliez pas d'expliquer les grandes lignes de votre solution qui peut éventuellement nécessiter de modifier vos précédents fichiers.

Livraison

– un fichier Java nommé `CEASTchampEcriture.java`
– un fichier C nommé `ilpField.c`

Notation sur 5 points

– 5 points