Examen réparti d'ILP

Christian Queinnec

10 janvier 2013

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.

Votre copie sera formée de fichiers textuels que vous laisserez aux endroits spécifiés dans votre espace de travail pour Eclipse. L'espace de travail pour Eclipse sera obligatoirement nommé `workspace` et devra être un sous-répertoire direct de votre répertoire personnel.

À l'exception des clés USB en lecture seule, tous les appareils électroniques communiquants sont prohibés (et donc notamment les téléphones portables). Vos oreilles ne doivent pas être reliées à ces appareils.

L'examen sera corrigé à la main, il est donc absolument inutile de s'acharner sur un problème de compilation ou sur des méthodes à contenu informatif faible. Il est beaucoup plus important de rendre aisé, voire plaisant, le travail du correcteur et de lui indiquer, par tout moyen à votre convenance, de manière claire, compréhensible et terminologiquement précise, comment vous surmontez cette épreuve. À ce sujet, vos fichiers n'auront que des lignes de moins de 80 caractères, n'utiliseront que le codage ASCII ou UTF-8 enfin, s'abstiendront de tout caractère de tabulation.

Le langage à étendre est obligatoirement ILP4.

Le paquetage Java correspondant sera nommé `fr.upmc.ilp.ilp4ar`. Sera ramassé, à partir de votre *workspace* (situé sous ce nom et directement dans votre répertoire HOME), le seul répertoire nommé `ILP/Java/src/fr/upmc/ilp/ilp4ar/` et tout ce qu'il contient.

Pour vous éviter de la taper à nouveau, voici l'url du site du master (mais il faut, pour y accéder, ne plus passer par le proxy) et celle de l'ARI où se trouvent de nombreuses documentations dont celle de Java :

```
http://www-master.ufr-info-p6.jussieu.fr/site-annuel-courant/
http://www-ari.ufr-info-p6.jussieu.fr/
```

Introduction

À l'instar du TME3, cet examen s'intéresse aux vecteurs mais dans le cadre exclusif d'ILP4. Il vous sera demandé d'ajouter quelques fonctions pour manipuler les vecteurs puis d'introduire une nouvelle syntaxe pour la création de vecteurs. Cette syntaxe usera de crochets, ainsi, on pourra écrire des programmes tels que :

```
let v = [ 1, 1+1, "ab" ]     // nouvelle syntaxe
in vectorSet(v, 0, vectorLength(v));
   print(vectorGet(v, 1-1)); // imprime 3
   print(vectorGet(v, 4/4)); // imprime 2
   if ( isVector(v) ) {
     vectorSet(v, 6/vectorLength(v), makeVector(3-1, pi));
     print(v); // pourrait imprimer [3, 2, [3.14, 3.14]]
   }
```

Cet examen ne s'intéressera pas à l'impression des vecteurs.

Question 1 – Interprétation (6 points)

Ajouter à l'interprète les fonctions suivantes, nommées exactement ainsi :

```
makeVector(taille, valeur)
isVector(valeur)
vectorLength(vecteur)
vectorGet(vecteur, index)
vectorSet(vecteur, index, valeur)
```

La fonction nommée `makeVector` prend une taille et une valeur et crée un vecteur de cette taille dont toutes les positions sont occupées par cette valeur. Le prédicat `isVector` ne rend vrai que si son argument est un vecteur. La fonction `vectorLength` retourne la taille de son argument si celui-ci est un vecteur. La fonction `vectorGet` prend un vecteur et un index et rend la valeur stockée à cet index dans le vecteur. La fonction `vectorSet` permet de modifier, dans un vecteur et à un index précis, la valeur stockée.

On rappelle qu'ILP est un langage sûr.

En revanche et à la différence de la solution fournie pour le TME3, la représentation des vecteurs d'ILP devra impérativement être `Object[]` c'est-à-dire des tableaux d'objets et non des instances de la classe `java.util.Vector`.

Remarque : le TME3 s'intéressait également à la fonction sinus, on débarrassera le code fourni de toute référence à sinus.

Livraison

- les fichiers en Java dans `workspace/ILP/Java/src/fr/upmc/ilp/ilp4ar/`
  (probablement, `VectorStuff.java` et `Process.java`)

Question 2 – Compilation (6 points)

On désire maintenant adjoindre ces mêmes fonctions au compilateur vers C. L'impression de vecteurs n'est pas demandée.

Livraison

- les fichiers C dans `workspace/ILP/Java/src/fr/upmc/ilp/ilp4ar/`
  (probablement `ilpVector.h`, `ilpVector.c` et `templateVector.c`)

Question 3 – Création de vecteur (8 points)

On souhaite disposer d'une syntaxe spécifique pour la création de vecteurs de taille quelconque et dont les divers éléments sont les valeurs des expressions entre crochets (cf. exemple ci-dessus).

Écrire une grammaire, nommée `grammar4vector.rnc`, pour prendre en compte cette nouvelle syntaxe. Écrire alors les classes Java nécessaires et l'éventuel schéma de compilation.

Livraison

- Dans le répertoire `workspace/ILP/Java/src/fr/upmc/ilp/ilp4ar/`, la grammaire `grammar4vector.rnc` et les fichiers Java associés nécessaires ainsi qu'un fichier `strategie.txt` expliquant comment vous traitez cette question.