Examen réparti de mi-semestre d'ILP

Christian Queinnec

12 novembre 2010

Directives

Le contrôle dure 3 heures.

Tous les documents sont autorisés, et notament ceux du cours. Vous disposez à ce titre d'un ordinateur avec accès http aux documents de cours et avec Eclipse pour vous permettre de rédiger les parties code de l'examen.

À l'exception des clés USB en lecture seule, tous les appareils électroniques sont **prohibés** (y compris les téléphones portables, les assistants numériques personnels et les agendas électroniques).

Le barème total est fixé à 20.

Votre travail pour l'examen sera fait dans l'espace de travail (*workspace*) Eclipse ILP dans lequel vous créerez le paquetage `fr.upmc.ilp.ilp2partiel2010` c'est-à-dire le répertoire `$HOME/workspace/ILP/Java/src/fr/upmc/ilp/ilp2partiel2010` qui contiendra les programmes (pas nécessairement compilables) que vous aurez écrits ainsi que le fichier textuel `partiel2010.txt` qui pourra contenir vos remarques additionnelles. Des consignes plus précises seront énoncées pour chaque question. Plus généralement, tout fichier ou répertoire, placé sous `$HOME/workspace/ILP/` et dont le nom contiendra `partiel2010` sera ramassé pour constituer votre copie. Rien d'autre ne sera ramassé !

Vos fichiers textuels seront codés en UTF-8, ils seront formés de lignes ne dépassant pas 80 caractères pour en faciliter la lecture.

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.

L'examen étant corrigé à la main, pour éviter de gâcher du papier, les copiés-collés abusifs tout autant qu'inutiles seront pénalisés.

Le langage à étendre est ILP2 ou ILP3.

1 Introduction

Un certain nombre de langages de programmation permettent d'introduire de nouvelles variables n'importe où dans une séquence. Ainsi, si en C89, on doit regrouper toutes les déclarations de variables locales au bloc en tête de ce bloc et donc écrire, par exemple :

```
{
  int i;
  float f;
  frobnicate(10);
  i = computeInteger();
  f = sin(i);
}
```

En C++, en Java, en Javascript, on peut par contre différer l'introduction des variables et ainsi rapprocher l'introduction de la variable et son initialisation ce qui peut s'écrire :

```
{
  frobnicate(10);
  int i;
  i = computeInteger();
  float f;
  f = sin(i);
}
```

On se propose d'ajouter cette caractéristique à ILP2.

Comme ILP2 est un langage non typé statiquement, on introduira un nouveau mot clé : `var` permettant d'introduire, dans toute suite d'instructions (ou séquence), une nouvelle variable locale dont la portée est le reste de la suite d'instructions (de cette même séquence). Ainsi l'exemple précédent pourrait-il aussi s'écrire en ILP2 comme à gauche ou à droite :

```
{
   frobnicate(10);
   var i;
   i = computeInteger();
   var f;
   f = sin(i);
}
```

```
{
  frobnicate(10);
  var i;
  var f;
  i = computeInteger();
  f = sin(i);
}
```

Attention, à la différence des blocs unaire ou n-aire qui introduisent des variables et les initialisent en même temps, le mot-clé `var` introduit une variable sans l'initialiser ! Ainsi donc peut-on aussi écrire :

```
{ var x;
  print(x);
}
```

ce qui doit donc conduire à une erreur puisque, si la variable `x` existe bien à l'invocation de `print`, elle n'a pas encore de valeur.

2 Questions

2.1 Grammaire (3 points)

Écrire une grammaire nommée `grammar2-partiel2010.rnc` définissant le mot clé `var` et permettant son usage au sein des seules séquences.

Nota: Vous pouvez ajouter en commentaire de cette grammaire, les remarques ou réflexions que vous jugez pertinentes.

2.2 Exemples (5 points)

Écrire au moins deux programmes en XML utilisant ce nouveau mot clé.

Nota: Vous pouvez ajouter en commentaire de ces programmes ce qu'ils sont sensés démontrer et calculer. Pensez à couvrir tous les cas d'utilisation, comment vous assurez-vous de les avoir tous couverts ?

2.3 Stratégie d'interprétation (6 points)

Décrivez, dans le fichier `~/workspace/ILP/Java/src/fr/upmc/ilp/ilp2partiel2010/partiel2010.txt`, votre stratégie d'interprétation de ce nouveau mot-clé.

Nota: Évitez de paraphraser la carte de référence (`http://www-master.ufr-info-p6.jussieu.fr/site-annuel-courant/Ext/queinnec/ILP/refcard.pdf`), appesantissez-vous sur les seuls points importants. Distinguez bien aussi ce qui est statique ou dynamique (relevant de la bibliothèque d'exécution). S'il y a une transformation de programme en jeu, donnez les schémas de réécriture (du même genre que les schémas de compilation vus en cours).

2.4 Stratégie de compilation (6 points)

Vous donnerez dans le fichier `~/workspace/ILP/Java/src/fr/upmc/ilp/ilp2partiel2010/partiel2010.txt` votre stratégie de compilation de ce nouveau mot-clé ainsi que les schémas de compilation de :

- la déclaration d'une variable avec le mot-clé `var`,
- la référence à une variable introduite avec `var`,
- l'affectation à une variable introduite avec `var`.

Nota: Là encore, évitez de paraphraser la carte de référence (`http://www-master.ufr-info-p6.jussieu.fr/site-annuel-courant/Ext/queinnec/ILP/refcard.pdf`), appesantissez-vous sur les seuls points importants ou demandés. Donnez les schémas de compilation.