Éléments de correction
Examen réparti de mi-semestre d'ILP

C.Queinnec

26 octobre 2009

Un corrigé est déjà disponible en ligne. Ce document n'ajoute que quelques remarques, plus ou moins générales, suite à la correction des copies.

1 Étape 1

Les programmes exemples et la grammaire (cf. étape 2) doivent être cohérents. Pour cela, vous avez appris en TME1 à valider un document XML par rapport à une grammaire RelaxNG.

Les exemples peuvent très souvent se contenter de reprendre les exemples textuels donnés dans l'énoncé. La grammaire doit au moins être capable de représenter tous les exemples de l'énoncé.

2 Étape 2

La grammaire doit être syntaxiquement correcte. Pour cela, vous avez appris en TME1 à valider une grammaire RelaxNG notamment en la traduisant la forme compacte en la forme XML.

La grammaire ne s'intéresse qu'à des concepts syntaxiques, textuels ou plus exactement à leur essence non encombrée de mots-clés. Les aspects sémantiques (typage, évaluation) ne sont pas de son ressort.

3 Étape 4

Les interfaces sont, la plupart du temps, entièrement générables par la grammaire. Il faut donc qu'interface et grammaire soient cohérentes. Attention à utiliser les bons types syntaxiques, principalement instruction ou expression.

4 Étape 5

Alors que ce qui précède est assez mécanique et sans beaucoup d'imagination, écrire les méthodes eval est la première question intéressante (au moins aux yeux des correcteurs), à ne pas omettre donc.

ILP est un langage non typé, il faut donc vérifier la nature des arguments de createRNG et generateRN avant de leur appliquer un traitement quelconque. Transtyper ne peut s'effectuer que si l'on est sûr de la nature de la valeur transtypée. Lorsque le résultat est un entier ILP, il doit être représenté par un BigInteger et non par un Integer.

Le tableau des générateurs aléatoires n'a pas à être dans une classe de l'AST mais plutôt dans le common. Il faut l'initialiser lorsque l'interprétation est demandée et pas au moment de l'analyse syntaxique. Le tableau doit être un singleton.

Une question aussi se posait : pouvait-on écrire createRNG(3.14) ? Autrement dit, createRNG prend-il comme argument un entier ou un nombre ?

Enfin, la méthode findFreeVariables était à écrire.

5 Étape 6

La fabrique ne posait aucun problème.

6 Étape 7

Les méthodes parse doivent bien sûr être cohérentes avec la grammaire. Là encore, elles pourraient être déduites automatiquement de la grammaire.

7 Étape 9

La destination a souvent été oubliée. Comme pour l'interprétation, il faut que la compilation vérifie que les valeurs fournies aux fonctions de la bibliothèque d'exécution sont bien des entiers. Mais comme cela nécessiterait de placer des instructions là où une expression est attendue, le mieux est de passer par une fonction (ou macro) intermédiaire qui effectuera la vérification.

8 Étape 11

Il fallait que Process impose la grammaire, l'analyseur et le patron C.