Éléments de correction
Examen final réparti d'ILP

Christian Queinnec

5 janvier 2012

Voici quelques éléments de correction de l'examen de janvier 2012 (cf. sujet enrichi). Si l'on prend le guide d'extension d'ILP3, la première question à se poser est « faut-il étendre la grammaire ?». La réponse, à mon sens, est plutôt oui car l'on introduit une nouvelle syntaxe avec une sémantique qui n'existait pas auparavant. Cette nouvelle syntaxe n'est pas une affectation comme le montre la variation de la 3ème question. La seconde question était de se demander si cette nouvelle syntaxe était une expression ou une instruction mais là vous aviez le choix et cela ne change en fait pas grand chose.

Une façon simple d'implanter cela tant à l'interprétation qu'à la compilation est illustrée par le schema suivant où la stipulation d'une valeur de retour de la fonction est transformée en une affectation sur une variable locale.

```
def f(...) {
   ...
     f := expr
   ...
}
```

```
def f(...) {
  let tmp = NULL
  in ...
     tmp = expr
     ...
     if tmp != NULL
     then tmp
     else erreur
}
```

Pour les questions 2 et 3, la question principale est « est-ce statique ou dynamique ? ». Cela peut être statique (le compilateur Java fait cette analyse) mais c'est difficilement faisable dans le temps de l'examen. En revanche, cela est simplissime dynamiquement. Pour la question 2, il suffit de vérifier, à la fin de la fonction, que la variable locale contenant la valeur de retour de la fonction a bien une valeur (cf. schéma précédent). Pour la question 3, il suffit de modifier la stipulation d'une valeur de retour pour vérifier qu'elle est bien dans l'état initial et non dans un état déjà initialisé.

Remarques

Lorsque vous répondez à une question, assurez-vous que vous traitez au moins, correctement, les exemples donnés dans l'énoncé ! Compter les stipulations de valeur de retour dans le corps d'une fonction et refuser toute fonction qui en contient plus d'une (`fact` par exemple) est erroné. Greffer en plus cette vérification dans `parse` (qui est déjà assez compliquée) est dangereux.

Il ne faut pas perturber l'ordre d'évaluation : déplacer, dans le schéma précédent, `expr` pour ne l'évaluer qu'en fin de fonction est faux. Compiler la stipulation de la valeur de retour avec un `return` est aussi faux car les instructions qui suivent ne seront alors pas exécutées.

Utiliser une variable globale pour le retour d'une fonction plutôt qu'une variable locale est dangereux : il faudrait ne jamais oublier de l'effacer avant toute invocation

(surtout en ILP3 avec les exceptions). Utiliser une variable locale de même nom que la fonction pour stocker la valeur de retour empêche toute récursion (comme dans `fact`).

Que votre code pour l'interprétation ne fasse pas de choses similaires à ce qui est fait en compilation devrait attirer votre attention. D'autre part, un programme, on l'interprète ou on le compile mais, dans la vraie vie, on ne l'interprète pas avant de le compiler : si on a déjà la réponse pourquoi diable la recalculer ?

Attention aux mélanges temporels ! À *parse-time*, on n'a pas accès à la valeur des expressions pas plus qu'à *compile-time*. De même le code C n'a pas accès à l'AST du programme ! Ne mélangez pas non plus les langages : `try` n'existe pas en C, qualifier une variable Java de `final` n'empêche pas une variable ILP d'être modifiée. Respectez également les langages : `return return;` n'est pas permis en C.

Quand vous modifiez du code existant, aidez le correcteur humain : signalez avec des commentaires les parties nouvelles ou supprimées ou modifiées. Présentez correctement vos programmes.

N'écrivez pas de tautologie du genre « on modifie parse() afin de résoudre le problème » sans autre explication. Cette phrase ne me convainc pas que vous savez résoudre le problème.

Comme rappelé dans les conditions générales de l'examen, soyez précis. Il y a je ne sais combien d'environnements dans ILP, parler d'« introduire un booléen dans l'environnement » ne veut rien dire : quel environnement ? Qui introduit le booléen ? Quand ? Assez souvent dans vos réponses, l'information introduite par un nœud de l'AST n'est pas forcément accessible depuis un autre nœud qui en aurait besoin et toutes ces informations ne sont pas nécessairement globales !