Éléments de correction
Examen final réparti d'ILP

Christian Queinnec

5 janvier 2012

Voici quelques éléments de correction de l'examen de novembre 2012, les codes correspondant sont dans Mercurial sous l'étiquette `partiel2012nov`.

Les nouvelles caractéristiques représentaient des expressions et non des instructions. Bien sûr il faut être cohérent, si l'on décidait que c'étaient des instructions, il ne fallait pas les compiler comme des expressions ! Et si la grammaire dit que les fils d'un noeud sont nommés `arg1` et `arg2`, il n'est pas question de les nommer `gauche` et `droit` dans le code.

Pour les tests, la difficulté portait principalement sur la vérification du court-circuit c'est-à-dire la vérification qu'une expression n'est pas calculée. Pour déterminer qu'une expression n'est pas calculée, une façon simple est que cette expression effectue une impression ce qui permet simplement de vérifier que l'impression n'a pas eu lieu. Verifier que `faux et print(..)` rend faux ne prouve rien sur le fait que `print()` n'a pas été évalué : c'est donc un test non prouvant.

Par ailleurs, si les tests permettent d'éprouver le code, il faut vérifier que les tests ne sont pas faux. Qu'un test ne soit correct que lorsqu'il imprime `erreur` est pour le moins étrange !

ILP, comme de nombreux langages, a une notion spécifique de ce qui est vrai ou faux. Ainsi en ILP (à la différence de C), `0` est vrai. Pour tester qu'une valeur ILP est vraie, il existe `ILP_isEquivalentToTrue` en C et `o != Boolean.FALSE` en Java. Il ne faut donc pas utiliser `o == Boolean.TRUE` qui ne capture pas tout ce qui vaut vrai.

Les « opérateurs » à court-circuit rendent comme valeur la première valeur calculée permettant de déterminer la valeur de l'expression toute entière. L'exemple le montrait (il faut toujours analyser les exemples) : `let x = 4 ET 6` initialise `x` avec 6 et non `true`.

Pour les schémas de compilation, voici quelques-unes des erreurs les plus criantes. Évaluer systématiquement les deux arguments avant de décider si un court-circuit était possible ! Évaluer le second argument avant le premier ! Évaluer deux fois le premier argument quand il vaut vrai ! Enfin, la destination est trop souvent oubliée.