Évaluation paresseuse

J. Sam, J.-C. Chappelier, V. Lepetit

version 1.0 de septembre 2013

L'évaluation d'une expression conditionnelle telle que :

```
(i != 0) and (25 / i > 12)
```

pose-t-elle problème à l'exécution si la variable `i` vaut 0 ?

On peut en effet se poser la question puisque si `i` vaut 0, le second opérande `(25 / i > 12)` de l'expression conditionnelle va causer une division par zéro s'il est évalué.

La réponse à cette question est en fait NON en raison de ce que l'on appelle *l'évaluation paresseuse* ("*lazy evaluation*" en anglais) : l'évaluation des opérandes se fait de la gauche vers la droite et seuls les opérandes nécessaires à la détermination de la valeur logique sont évalués. Dans le cas de notre exemple, comme l'opérande `(i != 0)` est faux, l'expression conditionnelle est fausse et `(25 / i > 12)` n'est pas évaluée.

De façon générale :

- pour une condition de la forme `X1 and X2 and ... and Xn`, les opérandes `Xi` ne sont évalués que *jusqu'au premier opérande faux* (s'il existe, auquel cas la condition est fausse, sinon elle est vraie) ;
- pour une condition de la forme `X1 or X2 or ... or Xn`, les opérandes ne sont évalués que *jusqu'au premier opérande vrai* (s'il existe, auquel cas la condition est vraie, sinon elle est fausse).

L'évaluation des conditions suivantes ne causera donc aucun problème à l'exécution même si `i` vaut 0 :

- `(i != 0) and (25 / i > 12)`
- `(i == 0) or (25 / i < 12)`