

# Composition d'Informatique

## Les Principes des Langages de Programmation (INF 321)

### Promotion 2011

Sujet proposé par Eric Goubault

12 juillet 2012

*Les questions qui suivent sont indépendantes et peuvent être traitées dans n'importe quel ordre. Les questions 2.d et 4.h sont optionnelles. Il est conseillé aux étudiants de lire rapidement tout le sujet avant de démarrer. Enfin, on attachera une grande importance à la clarté, à la précision et à la concision de la rédaction et des explications.*

On considère dans ce problème l'implémentation d'une feuille de calcul à la Excel. D'un point de vue logique, une feuille de calcul est un tableau bi-dimensionnel

$$C = (c_{i,j})_{i=0,\dots,n-1;j=0,\dots,m-1}$$

dont les éléments sont des formules arithmétiques ou des valeurs numériques (rentrées par un utilisateur) ou des cellules non définies (on utilise alors le symbole  $\perp$ ). Le but du problème est de donner une sémantique aux calculs sur cette feuille, et d'implémenter ces calculs en Java.

Pour fixer les idées, voici un exemple de feuille simple, avec  $n = 2$  et  $m = 2$ :

$$C = \begin{array}{|c|c|} \hline 1 & c_{1,1} \\ \hline \perp & c_{0,0} - c_{0,1} \\ \hline \end{array}$$

ainsi  $c_{0,0}$  est l'expression 1,  $c_{0,1}$  est l'expression  $c_{1,1}$ ,  $c_{1,0}$  est l'expression  $\perp$  et  $c_{1,1}$  est l'expression  $c_{0,0} - c_{0,1}$ .

## 1 Types de données et constructeurs

Les formules arithmétiques que l'on souhaite entrer dans chaque cellule ont pour grammaire:

$$AExpr ::= \perp \mid n \mid c_{i,j} \mid (AExpr) \mid AExpr + AExpr \mid \\ AExpr - AExpr \mid - AExpr \mid AExpr * AExpr \mid AExpr / AExpr$$

où  $n$  est une constante flottante ( $n \in Float$ , de type `float` donc),  $c_{i,j}$  dénote la valeur de la case numéro  $(i, j)$ ,  $i = 0, \dots, n - 1$ ,  $j = 0, \dots, m - 1$ . Ainsi, les expressions:

$$3.1415 \\ c_{1,2} \\ (c_{1,2} + c_{2,3}) * c_{0,0}$$

sont des expressions arithmétiques possibles, de même que les expressions apparaissant dans l'exemple  $C$  de l'introduction.

- (1.a) Définir une classe Java *concrète* `AExpr` qui implémente le type correspondant aux expressions arithmétiques (on ne cherchera pas forcément à être efficace en mémoire). On utilisera en particulier un champ `operator kind` où `operator` est le type énuméré (somme) suivant:

```
public enum operator {plus, times, div}
```

qui définit le type des opérateurs binaires.

- (1.b) Définir les constructeurs:

```
AExpr()
AExpr(float n)
AExpr(int i, int j)
AExpr(operator op, lexpr, rexpr)
AExpr(expr)
```

qui rendent respectivement  $\perp$ , un `AExpr` représentant le flottant  $n$ ,  $c_{i,j}$ , l'opération binaire `op` appliqué à une sous-expression gauche `lexpr` et à une sous-expression droite `rexpr`, et le moins appliqué à la sous-expression `expr`.

- (1.c) Définir une classe permettant de stocker une feuille de calcul. On l'appellera `Feuille`.

## 2 Sémantique et évaluation

On identifie, mathématiquement, une feuille de calcul  $C = (c_{i,j})$  à un "environnement"  $\rho_C : \mathbb{N} \times \mathbb{N} \rightarrow AExpr$  qui associe à chaque cellule  $(i, j)$ ,  $i = 0, \dots, n-1$ ,  $j = 0, \dots, m-1$ , l'expression arithmétique associée  $\rho_C(i, j) = c_{i,j}$ .

On dira que  $a \in AExpr$  contient la cellule  $c_{i',j'}$  (respectivement  $\perp$ ) si, par induction structurelle:

- $a$  est l'expression  $c_{i',j'}$  (respectivement  $\perp$ )
- $a$  est l'expression  $lexpr \text{ op } rexpr$  où  $\text{op} = +$  ou  $-$ ,  $*$ ,  $/$ , et  $lexpr$  et  $rexpr$  sont des expressions arithmétiques dont l'une au moins contient  $c_{i',j'}$  (respectivement  $\perp$ )
- $a$  est l'expression  $-lexpr$ , ou  $(lexpr)$  avec  $lexpr$  qui contient  $c_{i',j'}$  (respectivement  $\perp$ ).

On définit le graphe orienté de dépendance  $G_{\rho_C}$  d'une feuille  $C$  comme étant le graphe dont les nœuds sont les cellules  $c_{i,j}$  et  $\perp$ , et tel que l'on a un arc de  $c_{i,j}$  vers  $c_{i',j'}$  si  $\rho_C(i, j)$  est une expression contenant la cellule  $c_{i',j'}$ , et l'on a un arc de  $c_{i,j}$  vers  $\perp$  si  $\rho_C(i, j)$  est une expression contenant  $\perp$ .

On dira qu'une expression arithmétique  $a \neq \perp$  est *définie* dans l'environnement  $\rho_C$  si dans  $G_{\rho_C}$ :

- les cellules  $c_{i,j}$  définies dans  $a$  n'appartiennent pas à un cycle (orienté): ces cellules appartiennent donc à des chemins finis
- les nœuds finaux de ces chemins sont des cellules  $c_{i',j'}$ , jamais  $\perp$ , telles que  $\rho(c_{i',j'}) \in Float$

En termes simples, cela veut dire que toutes les cellules dont dépend  $a$  ne dépendent que de constantes numériques, et jamais de  $\perp$ , et ne sont pas dans un cycle (de dépendance) dans  $G_{\rho_C}$ .

Dans le cas de la feuille  $\mathcal{C}$  donnée en exemple du problème, le graphe  $G_{\rho_C}$  est:



et  $c_{0,0}$  est défini, mais pas  $c_{0,1}$ ,  $c_{1,0}$  ni  $c_{1,1}$ .

- (2.a) On considère dans cette question la feuille:

$$\mathcal{D} = \begin{array}{|c|c|} \hline c_{1,1} + c_{0,1} & 3 * c_{1,0} \\ \hline 2 & c_{0,0} + c_{0,1} \\ \hline \end{array}$$

Quelles en sont les cellules définies ? (On donnera pour ce faire le graphe  $G_{\rho_{\mathcal{D}}}$ ).

- (2.b) Écrire la sémantique dénotationnelle  $\llbracket expr \rrbracket$  des expressions arithmétiques  $a \in AExpr$ , par récurrence structurelle sur  $AExpr$ . On définira uniquement cette sémantique, dans le cas idéalisé où les calculs sont effectués dans les nombres réels (et non en précision finie). Cette fonction prend un environnement  $\rho$  et renvoie  $\llbracket a \rrbracket \rho$ , égal à la valeur (réelle) de l'expression  $a$  si  $a$  est définie,  $\perp$  sinon. On renverra  $\perp$  également en cas d'erreur dans une opération arithmétique (une division par zéro). Expliquer brièvement pourquoi cette récurrence structurelle est bien définie.

On cherche maintenant à implémenter la fonction sémantique de la question (2.b), que l'on nommera `Eval` telle que  $\llbracket a \rrbracket \rho = a.Eval(\rho)$ . Cette fonction implémentera directement la définition par récurrence structurelle de la question (2.b), en en faisant une fonction récursive.

- (2.c) Écrire le code de la méthode dynamique et récursive `Eval` de la classe `AExpr` prenant en argument une feuille  $C$ , et permettant d'évaluer l'expression sur laquelle elle s'applique. On supposera disposer d'une méthode `Defined`, que l'on ne cherchera pas à écrire, qui s'applique à une expression  $a$ , prend en argument une feuille  $C$ , et renvoie `true` si  $a$  est définie dans  $C$ , `false` sinon.
- (2.d - optionnel) Comment faire pour se passer de la méthode `Defined` pour programmer `Eval` ? Écrire le code correspondant.

*Indication:* On pourra s'aider d'un tableau de booléens `bool dejavu[n][m]` passé en argument à `Eval`, qui est initialisé à faux au moment de l'appel initial à `Eval`. L'entrée `dejavu[i][j]` est mis à vrai quand on a besoin d'évaluer (récursivement)  $c_{i,j}$ .

### 3 Gestion de la feuille de calcul

- (3.a) Écrire une méthode dynamique `Compute`, de la classe `Feuille`, qui évalue chaque cellule de la feuille courante (la fonction `Eval` du 2.c étant supposée donnée).
- (3.b) Que doit donner `Compute` sur  $\mathcal{C}$  ? et sur  $\mathcal{D}$  ?
- Le défaut de la méthode précédente est qu'une cellule peut être à  $\perp$  parce qu'elle n'est pas définie, mais aussi parce qu'une division par zéro a pu se produire. On souhaiterait maintenant pouvoir remonter l'information supplémentaire qu'une erreur de division par zéro s'est produite.
  - (3.c) Par quel mécanisme peut-on traiter simplement ces cas de division par zéro ?
  - (3.d) Dire comment modifier le code d'`Eval` (sans tout réécrire) de la question (2.c) et le code de `Compute` de la question (3.a) afin d'implémenter ce mécanisme.

### 4 Méthode alternative d'évaluation

De fait, une feuille de calcul pourrait également être vue comme un système d'équations de point fixe, donné par l'ensemble des expressions arithmétiques  $c_{i,j}$ . Ainsi, l'exemple de feuille simple  $\mathcal{C}$  donné en début de problème peut être aussi décrit par le système d'équations:

$$\begin{pmatrix} c_{0,0} \\ c_{0,1} \\ c_{1,0} \\ c_{1,1} \end{pmatrix} = F^{\mathcal{C}} \begin{pmatrix} c_{0,0} \\ c_{0,1} \\ c_{1,0} \\ c_{1,1} \end{pmatrix}$$

où

$$F^{\mathcal{C}} \begin{pmatrix} c_{0,0} \\ c_{0,1} \\ c_{1,0} \\ c_{1,1} \end{pmatrix} = \begin{pmatrix} 1 \\ c_{1,1} \\ \perp \\ c_{0,0} - c_{0,1} \end{pmatrix}$$

et  $F^C$  est croissante et continue (on ne cherchera pas à le démontrer) dans le CPO  $Float_{\perp}$  où  $\perp$  est le plus petit élément, et aucun couple  $x, y \in Float$  avec  $x \neq y$  n'est comparable. On notera  $F_{i,j}^C$  la composante de la fonction  $F^C$  correspondant à l'entrée  $(i, j)$ ,  $i = 0, \dots, n-1$ ,  $j = 0, \dots, m-1$ .

- (4.a) Quels sont les points fixes du  $F^C$  particulier défini ci-dessus ? De même pour  $F^D$  la fonctionnelle induite par la feuille  $\mathcal{D}$  de la question 2.a.
- (4.b) Quel est le plus petit point fixe de  $F^C$  ? de  $F^D$  ?
- (4.c) Donner les itérées successives du calcul de plus petit point fixe de  $F^C$  et de  $F^D$  par le théorème de Kleene.
- Prouver que, dans le cas général (donc pour tout  $F$  venant d'une feuille de calcul  $C$  quelconque):
  - (4.d) Si  $c_{i,j}$  n'est pas défini dans la feuille  $\rho$ , alors pour tout  $k \in \mathbb{N}$ ,  $(F^k)_{i,j}(\perp) = \perp$ , où  $(F^k)_{i,j}$  est la composante  $(i, j)$  de l'itérée  $k$ ième de  $F$
  - (4.e) Si tous les  $c_{i,j}$  sont définis dans la feuille  $\rho$ , on peut définir la profondeur de dépendance  $k_{i,j}^a \in \mathbb{N}$  de toute expression  $a$  par rapport à la cellule  $c_{i,j}$  dans l'environnement  $\rho_C$  comme étant la longueur maximale d'un chemin dans  $G_{\rho_C}$  partant des cellules contenues dans  $a$  et arrivant en  $c_{i,j}$ .  
Appelons  $C^1$  l'ensemble des cellules  $c_{i,j}$  telles que  $\rho(c_{i,j})$  est une constante numérique. Prouver maintenant que si tous les  $c_{i,j}$  sont définis dans la feuille  $\rho$ ,  $F^k(\perp)$ ,  $k \geq 1$ , donne la valeur numérique (donc différente de  $\perp$ , au sens de la sémantique dénotationnelle de la question (2.a)) de l'ensemble  $C^k$  des cellules  $c_{i,j}$  telles que  $k > k_{i',j'}^{c_{i,j}}$  pour toutes cellules  $c_{i',j'} \in C^1$ .
  - (4.f) En déduire que le plus petit point fixe de la fonctionnelle  $F$  est égal à la sémantique dénotationnelle donnée à la question (2.a).
- (4.g) Écrire une version itérative de la méthode *Eval* de la question (2.c), implémentant l'itération de Kleene, que l'on sait donner le même résultat que la méthode récursive de la question (2.c), grâce à la question (4.f) (on n'a pas besoin d'avoir répondu à (4.f) pour faire cette question).
- (4.h - optionnel) Soit  $P$  le prédicat suivant sur les  $a \in AExpr$  et les feuilles  $\rho$ :  $P(a, \rho)$  est vrai si  $a$  est définie dans  $\rho$ . Donner les assertions nécessaires à la preuve à la Hoare du code itératif d'Eval de la question (4.d), à chaque ligne du code, permettant de prouver

$$\{\neg P(a, \rho)\} \mathbf{x=a. Eval}(\rho) \{x = \perp\}$$

Vérifier les assertions de preuve en utilisant les règles de preuve à la Hoare du cours. On pourra bien sûr s'inspirer de la preuve à la question (4.d).