Validation d'algorithmes
L3 MIAGE, UGA
2020/2021

raphael.charrondiere@inria.fr

Examen de Validation d'algorithmes

Modalités : Cet examen suit les règles usuelles, aucun document n'est autorisé, vous avez 1h30 pour répondre.

Pour chaque question, une indication sur le temps maximal à investir est indiquée.

Important :

- Pensez à vous relire.
- Soyez clair, soignez la présentation et votre écriture. Un texte bref mais précis est plus utile qu'un discours long et confus.
- Justifiez vos réponses.
- Faites des phrases en français au lieu d'utiliser les symboles mathématiques.
- Ne perdez pas de temps mais lisez le sujet en entier avant de commencer, ensuite, répondez aux questions qui vous semblent facile en premier.
- Si vous ne connaissez pas la réponse inutile d'écrire n'importe quoi, ça se verra et le correcteur sera moins conciliant.

Rappel de cours

On rappelle d'abord les règles d'inférence pour la sémantique à grands pas de IMP.

$$\frac{}{\sigma, \textbf{skip} \Downarrow \sigma} \qquad \frac{\sigma, c_1 \Downarrow \sigma' \quad \sigma', c_2 \Downarrow \sigma''}{\sigma, c_1 \,\textbf{;} c_2 \Downarrow \sigma''} \qquad \frac{\sigma, a \looparrowright k}{\sigma, x \,\textbf{:=} a \Downarrow \sigma'}\ \sigma' = \sigma[k/x]$$

$$\frac{\sigma, b \looparrowright true \quad \sigma, c_1 \Downarrow \sigma'}{\sigma, \textbf{if b then } c_1 \textbf{ else } c_2 \Downarrow \sigma'} \qquad \frac{\sigma, b \looparrowright false \quad \sigma, c_2 \Downarrow \sigma'}{\sigma, \textbf{if b then } c_1 \textbf{ else } c_2 \Downarrow \sigma'}$$

$$\frac{\sigma, b \looparrowright false}{\sigma, \textbf{while b do } c \Downarrow \sigma} \qquad \frac{\sigma, b \looparrowright true \quad \sigma, c \Downarrow \sigma' \quad \sigma', \textbf{while b do } c \Downarrow \sigma''}{\sigma, \textbf{while b do } c \Downarrow \sigma''}$$

On rappelle la syntaxe des expressions algébriques :

- AB : A puis B
- A|B : A ou B
- A* : A plusieurs fois (potentiellement zéro fois)
- A+ : A une fois ou plus, soit AA*
- ϵ : l'expression vide (si besoin)
- A ? : A zéro ou une fois, soit (A|ϵ)
- (A) : utilisez des parenthèses, les opérateurs ne se rapportent qu'au dernier élément, exemple AB* veut dire A suivi de plusieurs B, mais (AB)* veut dire AB répété plusieurs fois.

Partie 1: Sémantique opérationnelle

Question 1.1 : Restitution de connaissances *(5 minutes)* . . . . . **1 Point**
Expliquez ce qu'est un arbre de dérivation.

Question 1.2 : Règles d'inférences *(10 minutes)* . . . . . **3 Points**
On enrichie IMP avec de nouvelles commandes présentées si dessous, pour chacune d'entre elles, donnez les règles d'inférence correspondantes.

`c` est une commande IMP, `x` est une variable, `b` est une condition (expression booléenne), `i` et `j` sont des expressions arithmétiques.

a - `do c while b`. Cette commande exécute l'instruction `c` puis recommence tant que `b` est vrai.

b - `x:=b?i:j`. Cette commande évalue la condition `b` puis attribue à `x` la valeur `i` si `b` est vraie, `j` sinon.

Attention piège : pensez que `i` et `j` sont des expressions arithmétiques et non des entiers.

Partie 2: Logique de Hoare

$\{N \geqslant 0\} \implies \{\dots\}$
$R := 0$
$\{\dots\}$
$X := 0$
$\{\dots\}$
while$(X \geqslant N)$
  $\{\dots\} \implies \{\dots\}$
  $R := R + X$
  $\{\dots\}$
  $X := X + 1$
  $\{\dots\}$
$\{\dots\} \implies \left\{R = \frac{N(N+1)}{2}\right\}$

Classiquement pour prouver un programme avec la logique de Hoare, on doit annoter un programme, c'est à dire mettre des prédicats un peu partout entre les instructions comme montré sur l'exemple à gauche.

Question 2.1 : Une question de vocabulaire *(2 minutes)* . . . . . **1 Point**
Le premier et le dernier prédicat ($N \geqslant 0$ et $R = \frac{N(N+1)}{2}$) sont particuliers, comment nomme-t-on ce couple dans Verifast.

Question 2.2 : Invariants *(5 minutes)* . . . . . **1 Point**
Vous avez du remarquer une différence notable entre les invariants de Verifast et ceux vu en cours, pouvez-vous l'expliquer ?

Question 2.3 : Variants *(5 minutes)* . . . . . **2 Points**
Les variants dans Verifast sont assez basiques, il s'agit juste d'une expression arithmétique dans l'ensemble des entiers. Pour enrichir la notion de variant, dans le cours, on a utilisé un ensemble arbitraire muni d'une relation d'ordre $\leqslant$ bien fondé, pouvez-vous expliquer le besoin d'une telle relation en deux lignes environ ?

Partie 3: Boite blanche

La fonction suivante est la méthode d'une classe écrite en C++. Vous n'avez pas besoin de la comprendre pour répondre aux questions. Il s'agit ici, de chercher un ensemble de tests à effectuer.

```
1  Motif Factory::computeMotif(size_t id)
2  {
3    if(renders.find(id)!=renders.end())
4      return renders[id];
5    int depth=0;
6    std::stack<Action*> stack;
7    stack.push(mng->get(id));
8    Motif mwork;
9    while(true)
10   {
11     depth++;
12     Action* da = stack.top();
13     if(da==nullptr)break;
14     if(renders.find(id)!=renders.end())
15     {
16       mwork=renders[id];
17       break;
18     }
19
20     stack.push(da->Prev());
21   }
22   if(stack.top()==nullptr)
23     stack.pop();
24   while(!stack.empty())
25   {
26     stack.top()->apply(mwork);
27     stack.pop();
28   }
29   if(depth>=cacheDepth3)
30     renders[id]=mwork;
31   return mwork;
32 }
```

Question 3.1 : Graphe de flot de contrôle *(15 minutes)* . . . . . **3 Points**
Veuillez tracer le graphe de flot de contrôle correspondant à la fonction.

Important : les nœuds de décisions doivent avoir une forme différente des nœuds de commandes pour pouvoir les reconnaitre facilement, et vous **devez** étiquetez les arcs sortant avec la valeur de la condition.

Indication 1 : On considère les appels de fonctions comme des simples instructions.

Indication 2 : Pour les nœuds de commandes **uniquement**, vous pouvez écrire seulement les numéros de ligne. Pour tout nœud de condition, **recopiez** clairement la condition.

Indication 3 : prévoyez une page entière pour le graphe.

Question 3.2 : Chemins *(5 minutes)* . . . . . **1 Point**

Commencez par donner des lettres aux nœuds de votre graphe pour les identifier, puis indiquez l'expression algébrique de tout les chemins du graphe. *Vous pouvez vous référer au rappel en début de sujet pour la syntaxe des expressions régulières.*

Question 3.3 : Tests *(12 minutes)* . . . . . **3 Points**

En le **justifiant** :

a - proposez un ensemble de chemins pour le critère «tout les nœuds».

b - proposez un ensemble de chemins pour le critère «tout les arcs».

c - combien de chemins faut-il pour couvrir une base de chemins indépendants ?

Partie 4: Boite noire

On rappelle que les tests en boite noire sont des tests qui ne considèrent pas le code, d'ailleurs celui n'est peut-être pas accessible au testeur.

On suppose avoir une structure de données multi-ensemble d'entiers. Elle représente un ensemble qui contient des entiers, mais un entier peut être présent plusieurs fois dans cet ensemble. Par exemple l'ensemble peut contenir une fois 1, deux fois 2 et trois fois 3, on l'écrit par $\{1, 2, 2, 3, 3, 3\}$.

Cette structure contient une fonction d'affichage que l'on va supposer correcte, le testeur l'appellera pour voir le contenu de la structure de données. Aussi, on supposera avoir à disposition une fonction qui crée un multi-ensemble vide.

Question 4.1 : Ajout d'élément *(10 minutes)* . . . . . **2 Points**

On voudrait vérifier le fonctionnement de la fonction `ajouterElement` qui ajoute un élément (donc un entier) à la structure, proposez un ensemble de tests en justifiant (expliquant) votre choix.

Indice : S'il y a besoin d'un ensemble précis, il suffit de partir de l'ensemble vide, et d'ajouter un à un les éléments.

Partie 5: Automates de Mealy

Question 5.1 : Le pourquoi *(10 minutes)* . . . . . **3 Points**

Commencez par donner une définition très rapide au automates de Mealy, puis donnez une utilisation de cet outil dans le cadre de ce cours. Finalement, un automate de Mealy doit-il plutôt être catégorisé parmi les tests boîte blanche ou boîte noire ?

Total : 20 points, temps estimé : 79 minutes