Université de Paris
M2 – DM de Méthodes Formelles pour la Vérification

Novembre 2020

NB: Toutes les réponses doivent être justifiées et clairement rédigées.

Exercice 1

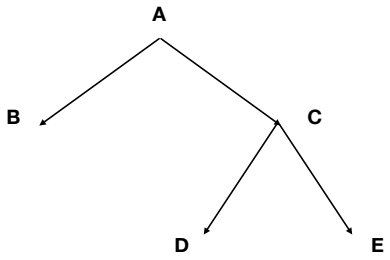
Prouver formellement la correction de la fonction de tri par extraction du minimum vue en cours.

Indication: On ne demande pas d'écrire la fonction d'extraction du minimum. On peut utiliser sa spécification formelle donnée dans le cours.

Exercice 2

On considère le type des arbres binaires (non vides) défini par les deux constructeurs L et N :

$$\begin{aligned} L &: \star \rightarrow BT[\star] \\ N &: \star \times BT[\star] \times BT[\star] \rightarrow BT[\star] \end{aligned}$$



Par exemple, le terme $N(A, L(B), N(C, L(D), L(E)))$ correspond à l'arbre représenté dans la figure ci-contre.

On considère sur les éléments d'un arbre binaire les trois ordres standards : infixe, pré-fixe, et post-fixe. Par exemple, pour l'arbre donné dans la figure, ces ordres correspondent à :

- Infixe : B A D C E
- Pré-fixe : A B C D E
- Post-fixe : B D E C A

Question 1: Pour chacun des trois ordres infixe, pré-fixe, et post-fixe, définir une fonction qui, étant donné un arbre binaire quelconque t, élément du type $BT[\star]$ introduit ci-dessus, produit une liste (élément du type $List[\star]$) dont les éléments sont ceux de l'arbre t, listés dans l'ordre considéré.

On considère maintenant que le domaine des éléments des arbres est muni d'une relation d'ordre. Un arbre binaire de recherche (BST) est alors un arbre tel que, pour chaque noeud, les éléments dans le sous-arbre gauche sont inférieurs ou égaux à l'élément qui se trouve au noeud, et ce dernier est inférieur strictement à tous les éléments du sous-arbre droit.

Question 2: Prouver formellement que pour tout arbre binaire de recherche, la liste de ses éléments dans l'ordre infixe est ordonnée.

Indication: Utiliser une définition formelle de la propriété d'être un BST exprimée de manière inductive.

Question 3: Définir une fonction de tri d'une liste quelconque qui utilise les BST comme représentation intermédiaire. Donner une preuve formelle de la correction de cette fonction par rapport à la spécification d'une fonction de tri.

Indication: On peut supposer que l'on dispose d'une fonction d'insertion d'un élément dans un arbre binaire de recherche (BST), et utiliser sa spécification formelle.

Exercice 3

Pour chaque triplet de Hoare ci-dessous, soit prouver qu'il est valide en donnant la preuve de sa validé dans la logique de Hoare, soit prouver sa non validité en donnant un contre-exemple. Dans le dernier cas, donner une nouvelle pré-condition, qui soit la plus générale permettant d'obtenir un triplet de Hoare valide tout en gardant la même post-condition du triplet d'origine.

1. $\{x = 2\}\ x := x + 1\ \{x = 3\}$
2. $\{y \geq 2\}\ x := y\,;\ y := -1\ \{x > 0\}$
3. $\{true\}$ `if` $true$ `then` $y := x$ `else` $y := 2 * x\,;\ x := y - 1\ \{x > 0\}$
4. $\{y > 0\}$ `if` $y > 0$ `then` $x := y$ `else` $x := -y\ \{x > 0\}$
5. $\{true\}$ `if` $y > 0$ `then` $x := y$ `else` $x := -y\ \{x > 0\}$

Exercice 4

Pour chacun des programmes ci-dessous, donner une spécification formelle ainsi qu'une preuve de correction partielle dans la logique de Hoare.

$r, z : \mathbb{N}$;
Fun1 $(x, y : \mathbb{N})\ =$
    `assume`$(x \geq 0 \wedge y \geq 0)$;
    $r := 0$;
    $z := x$;
    `while` $z \neq 0$ `do`
        $r := r + y$
        $z := z - 1$

$r, i, j : \mathbb{N}$;
Fun2 $(x, y : \mathbb{N})\ =$
    `assume`$(x \geq 0 \wedge y \geq 0)$;
    $r := 0$;
    $i := 0$;
    `while` $i \neq x$ `do`
        $j := 0$;
        {`while` $j \neq y$ `do`
            $r := r + 1$;
            $j := j + 1$ } ;
        $i := i + 1$

Exercice 5

On définit la fonction Rac qui calcule la racine carrée d'un entier naturel :

$c, s : \mathbb{N}$;
Rac $(n : \mathbb{N})$ =
    `assume`$(n \geq 0)$;
    $c := 0$;
    $s := 1$;
    `while` $s \leq n$ `do`
        $c := c + 1$
        $s := s + 2 * c + 1$
    `assert`$(c \geq 0 \wedge c^2 \leq n < (c+1)^2)$

Question 1: Prouver dans la logique de Hoare la correction partielle du programme Rac, c'est-à-dire que, en supposant que le paramètre n est un entier positif, si le programme s'arrête, alors l'assertion $c \geq 0 \wedge c^2 \leq n < (c+1)^2$ est vraie à la fin de l'exécution du programme.

Indication: Exprimer le lien entre s et c (réflechir à la question : "que calcule s ?") au cours des itérations de la boucle, et l'utiliser pour définir un invariant qui, à la sortie de la boucle impliquera la post-condition désirée.

Question 2: Prouver formellement que le programme Rac termine.

Indication: Donner une fonction de rang (ranking function) qui associe à chaque état du programme un entier naturel, et dont la valeur décroit strictement à chaque exécution du corps de la boucle, et l'utiliser dans la règle de l'itération de la logique de Hoare pour prouver la correction totale.