

## Preuves de programmes – TP n° 7

## Logique de Hoare

### Problème (extrait de l'examen de février 2006)

Dans ce problème, on se place en logique du premier ordre (multi-sortée) à deux sortes : la sorte *int* des entiers relatifs (munie des structures habituelles), et la sorte *tab* des tableaux d'entiers. Les tableaux sont manipulés aux moyen de trois symboles de fonction :

- $\text{len}(tab) : int$  (longueur d'un tableau)
- $\text{acc}(tab, int) : int$  (accès à un élément d'un tableau)
- $\text{st}(tab, int, int) : tab$  (affectation d'un élément d'un tableau).

munis des axiomes suivants :

$$\begin{aligned} \forall t \forall i \forall v [(0 \leq i < \text{len}(t)) \Rightarrow \text{len}(\text{st}(t, i, v)) = \text{len}(t)] \\ \forall t \forall i \forall v [(0 \leq i < \text{len}(t)) \Rightarrow \text{acc}(\text{st}(t, i, v), i) = v] \\ \forall t \forall i \forall j \forall v [(0 \leq i < \text{len}(t)) \wedge (0 \leq j < \text{len}(t)) \wedge (i = j) \Rightarrow \text{acc}(\text{st}(t, i, v), j) = \text{acc}(t, j)] \end{aligned}$$

Dans ce qui suit, on utilise les abréviations  $t[i] \equiv \text{acc}(t, i)$  et  $t[i] := v \equiv t := \text{st}(t, i, v)$ .

**L'instruction swap** On aimerait ajouter au langage de commandes une instruction  $\text{swap}(t, i, j)$  qui permute deux éléments  $i$  et  $j$  d'un tableau  $t$ . Pour cela, on introduit dans le langage logique un nouveau symbole de fonction ternaire  $\text{exch}$ , où  $\text{exch}(t, i, j)$  représente le tableau  $t$  dans lequel les éléments d'indice  $i$  et  $j$  ont été permutés.

1. En vous inspirant des axiomes ci-dessus, donner les 4 axiomes de la fonction  $\text{exch}$ . Vérifier que ces axiomes sont satisfaits si on définit  $\text{exch}(t, i, j)$  comme la macro suivante :

$$\text{exch}(t, i, j) \equiv \text{st}(\text{st}(t, i, \text{acc}(t, j)), j, \text{acc}(t, i))$$

2. Que peut on dire de  $\text{exch}(t, i, j)$  si l'un des indices  $i$  et  $j$  est en dehors des bornes de  $t$ ?

Dans ce qui suit, l'instruction  $\text{swap}$  est définie par l'abréviation :  $\text{swap}(t, i, j) \equiv t := \text{exch}(t, i, j)$ .

**Calcul de préconditions** On considère les deux morceaux de code suivants :

$$\begin{aligned} C &\equiv \text{if } p < q \text{ then begin swap}(t, p, q); p := p + 1; q := q - 1 \text{ end} \\ D &\equiv \text{while } p \leq q \wedge t[p] < \text{piv} \text{ do } p := p + 1 \text{ done} \end{aligned}$$

3. Calculer la précondition la plus faible  $\mathbf{WP}(C, Q)$  de la commande  $C$  pour une postcondition  $Q$  arbitraire. Quelles sont les obligations de preuves produites ?
4. Même question pour la commande  $D$ , en supposant que la boucle  $\text{while}$  est annotée avec un invariant  $I$  et un variant  $V$  arbitraires. Quelle variant  $V$  serait-il raisonnable de prendre ici ?

**Partitionnement dans l'algorithme de *quick-sort*** On considère le programme de partitionnement suivant, extrait d'un algorithme de quick-sort :

```

{len(t) ≥ 1}
  p := 0;
  q := len(t) - 1;
  while① p ≤ q do
    while② p ≤ q ∧ t[p] < piv do p := p + 1 done;
    while③ p ≤ q ∧ t[q] ≥ piv do q := q - 1 done;
    if p < q then begin swap(t, p, q); p := p + 1; q := q - 1 end
  done
{(0 ≤ p ≤ len(t)) ∧ ∀i(0 ≤ i < p ⇒ t[i] < piv) ∧ ∀i(p ≤ i < len(t) ⇒ t[i] ≥ piv)}

```

où  $p$ ,  $q$  et  $piv$  sont des variables entières, et  $t$  une variable de tableau d'entiers.

5. Proposer des invariants et des variants raisonnables pour chacune des trois boucles ①, ② et ③.
6. On désigne par  $I_1$ ,  $I_2$ ,  $I_3$  (resp.  $V_1$ ,  $V_2$ ,  $V_3$ ) les invariants (resp. les variants) associés aux boucles ①, ② et ③. Écrire les obligations de preuve engendrés par la méthode de preuve de correction à l'aide des préconditions les plus faibles.
7. Vérifiez que ces obligations sont satisfaites avec les invariants que vous avez proposés à la question 5.