Assignment 2: Verification

Maxwell Bo 43926871

April 13, 2017

1 Part A

Let

$$\begin{aligned} pre \quad &\triangleq \quad D.len \geqslant max(\{A.len,\ B.len,\ C.len\}) \\ &\qquad \wedge\ sorted(A)\ \wedge\ sorted(B)\ \wedge\ sorted(C) \end{aligned}$$

$$post(r) \quad \triangleq \quad D_{[0,r)} = A \cap B \cap C$$

$$\begin{aligned} & i,j,k,r,D : [pre, post(r)] \\ \sqsubseteq\ & \{\text{Composition: middle predicate is } inv\} \\ & i,j,k,r,D : [pre, inv];\ i,j,k,r,D : [inv, post(r)] \end{aligned}$$

where

$$\begin{aligned} inv \quad &\triangleq \quad D_{[0,r)} = A_{[0,i)} \cap B_{[0,j)} \cap C_{[0,k)} \\ &\qquad \wedge\ r \in [0, D.len]\ \wedge\ i \in [0, A.len]\ \wedge\ j \in [0, B.len]\ \wedge\ k \in [0, C.len] \end{aligned}$$

$$\begin{aligned} \sqsubseteq\ & \{\text{Assignment: } pre \Rrightarrow inv[i,j,k,r \backslash 0,0,0,0]\} \\ & i,j,k,r := 0,0,0,0;\ i,j,k,r,D : [inv, post(r)] \end{aligned}$$

$\because$

$$\begin{aligned} inv[i,j,k,r\backslash 0,0,0,0] \quad &\equiv \quad D_{[0,0)} = A_{[0,0)} \cap B_{[0,0)} \cap C_{[0,0)} \\ &\qquad \wedge\ 0 \in [0, D.len]\ \wedge\ 0 \in [0, A.len]\ \wedge\ 0 \in [0, B.len]\ \wedge\ 0 \in [0, C.len] \\ &\equiv \quad \varnothing = (\varnothing \cap \varnothing \cap \varnothing)\ \wedge\ (\text{true}\ \wedge\ \text{true}\ \wedge\ \text{true}\ \wedge\ \text{true}) \\ &\equiv \quad \varnothing = \varnothing\ \wedge\ \text{true} \\ &\equiv \quad \text{true} \end{aligned}$$

$$\begin{aligned} \sqsubseteq\ & \{\text{Strengthen post: } inv \wedge \neg guard \Rrightarrow post(r)\} \\ & i,j,k,r := 0,0,0,0;\ i,j,k,r,D : [inv, inv \wedge \neg guard] \end{aligned}$$

where *guard* is a function that takes i, j, k as implicit parameters, s.t.

$$guard(i,j,k) \quad \triangleq \quad (i \neq A.len \vee j \neq B.len \vee k \neq C.len)$$

$\therefore$

$$inv \wedge \neg guard \quad \equiv \quad inv \wedge (i = A.len \wedge j = B.len \wedge k = C.len)$$

Assuming $(i = A.len \ \wedge \ j = B.len \ \wedge k = C.len)$ holds, we can show that still

$$inv \wedge (i = A.len \ \wedge \ j = B.len \ \wedge k = C.len) \ \Rrightarrow \ post(r)$$

$\because$

$$\begin{array}{rcl} inv \wedge (i = A.len \ \wedge \ j = B.len \ \wedge k = C.len) & \equiv & inv[i,j,k \backslash A.len, B.len, C.len] \\ & \equiv & D_{[0,r)} = A_{[0,A.len)} \cap B_{[0,B.len)} \cap C_{[0,C.len)} \\ & & \wedge \ r \in [0, D.len] \ \wedge \ A.len \in [0, A.len] \ \wedge \ B.len \in [0, B.len] \ \wedge \ C.len \\ & \equiv & (D_{[0,r)} = A \cap B \cap C) \ \wedge \ (r \in [0, D.len] \ \wedge \ \text{true} \ \wedge \ \text{true} \ \wedge \ \text{true}) \\ & \equiv & (D_{[0,r)} = A \cap B \cap C) \ \wedge \ (r \in [0, D.len]) \end{array}$$

$$\begin{array}{rcl} (D_{[0,r)} = A \cap B \cap C) \ \wedge \ (r \in [0, D.len]) & \Rrightarrow & post(r) \\ & \Rrightarrow & D_{[0,r)} = A \cap B \cap C \end{array}$$

$\sqsubseteq$ {Repetition}
$i, j, k, r := 0, 0, 0, 0;$
do $(i \neq A.len \ \vee \ j \neq B.len \ \vee \ k \neq C.len) \rightarrow$
$\quad i, j, k, r, D : [inv \ \wedge \ guard, \ inv \ \wedge \ (0 \leqslant V < V_0)]$
od

where

$$\begin{array}{rcl} V & \triangleq & (A.len - i) + (B.len - j) + (C.len - k) \\ & \triangleq & (A.len + B.len + C.len) - (i + j + k) \end{array}$$

$\not\sqsubseteq$ {Selection: $inv \ \wedge \ guard \not\Rrightarrow (G_1(i,j) \vee G_2(j,k) \vee G_3(k,i) \vee G_4(i,j,k))$}
$i, j, k, r := 0, 0, 0, 0;$
do $(i \neq A.len \ \vee \ j \neq B.len \ \vee \ k \neq C.len) \rightarrow$
$\quad$ **if** $(A_i > B_j) \rightarrow i, j, k, r, D : [(A_i > B_j) \wedge inv \wedge guard, \ inv \wedge (0 \leqslant V < V_0)]$
$\quad [\!] \ (B_j > C_k) \rightarrow i, j, k, r, D : [(B_j > C_k) \wedge inv \wedge guard, \ inv \wedge (0 \leqslant V < V_0)]$
$\quad [\!] \ (C_k > A_i) \rightarrow i, j, k, r, D : [(C_k > A_i) \wedge inv \wedge guard, \ inv \wedge (0 \leqslant V < V_0)]$
$\quad$ **fi** $(A_i = B_j) \wedge (B_j = C_k) \rightarrow i, j, k, r, D : [(A_i = B_j) \wedge (B_j = C_k) \wedge inv \wedge guard, \ inv \wedge (0 \leqslant V < V_0)]$
od

where

$$\begin{array}{rcl} G_1(i,j) & \triangleq & A_i > B_j \\ G_2(j,k) & \triangleq & B_j > C_k \\ G_3(k,i) & \triangleq & C_k > A_i \\ G_4(i,j,k) & \triangleq & (A_i = B_j) \wedge (B_j = C_k) \\ G_{union}(i,j,k) & \triangleq & (G_1(i,j) \vee G_2(j,k) \vee G_3(k,i) \vee G_4(i,j,k) \end{array}$$

$\therefore$

$G_1(i,j) \vee G_2(j,k) \vee G_3(k,i) \vee G_4(i,j,k)$
$\equiv$ {Expansion of the guard definitions}
$(A_i > B_j) \vee (B_j > C_k) \vee (C_k > A_i) \vee ((A_i = B_j) \wedge (B_j = C_k))$
$\equiv$ {Transitivity}
$(A_i > B_j) \vee (B_j > C_k) \vee (C_k > A_i) \vee (A_i = B_j = C_k)$
$\equiv$ {Redefinition}
$G_{union}(i,j,k)$

Actions that have undefined behaviour, such as out-of-bounds array indexing, are inexpressible in the Guarded Query Language. Therefore, for any array-index pairing A_i, there is an implicit constraint that $i \in [0, A.len)$.

Thus

$$\begin{aligned} & \equiv \quad \{\text{Implicit constraints}\} \\ & \quad ((A_i > B_j) \vee (B_j > C_k) \vee (C_k > A_i) \vee (A_i = B_j = C_k)) \ \wedge \ (i \in [0, A.len) \ \wedge \ j \in [0, B.len) \ \wedge \ k \in [0, C.len)) \end{aligned}$$

With these new implicit constraints, we can conclude that

$$\begin{aligned} inv \wedge guard \quad \not\Rightarrow \quad & ((A_i > B_j) \vee (B_j > C_k) \vee (C_k > A_i) \vee (A_i = B_j = C_k)) \\ & \wedge \ (i \in [0, A.len) \ \wedge \ j \in [0, B.len) \ \wedge \ k \in [0, C.len)) \end{aligned}$$

For convenience

$$implicit\_constraints(i, j, k) \quad \triangleq \quad i \in [0, A.len) \ \wedge \ j \in [0, B.len) \ \wedge \ k \in [0, C.len)$$

By counter-example, let $i, j, k, r = A.len, 0, 0, 0$, and choosing $A = \{0\}$, $B = \{1\}$, $C = \{1\}$, s.t. $A.len = 1$, $B.len = 1$ and $C.len = 1$.

$$\begin{aligned} inv[i, j, k, r \backslash A.len, 0, 0, 0] \quad \equiv \quad & D_{[0,0)} = A_{[0,A.len)} \cap B_{[0,0)} \cap C_{[0,0)} \\ & \wedge 0 \in [0, D.len] \ \wedge \ A.len \in [0, A.len] \ \wedge \ 0 \in [0, B.len] \ \wedge \ 0 \in [0, C.len] \\ \equiv \quad & \varnothing = (A \cap \varnothing \cap \varnothing) \ \wedge \ (\text{true} \ \wedge \ \text{true} \ \wedge \ \text{true} \ \wedge \ \text{true}) \\ \equiv \quad & \varnothing = \varnothing \ \wedge \ \text{true} \\ \equiv \quad & \text{true} \end{aligned}$$

$$\begin{aligned} guard[i, j, k, r \backslash A.len, 0, 0, 0] \quad \equiv \quad & (i \neq A.len \ \vee \ j \neq B.len \ \vee \ k \neq C.len)[i, j, k, r \backslash A.len, 0, 0, 0] \\ \equiv \quad & (A.len \neq A.len \ \vee \ 0 \neq B.len \ \vee \ 0 \neq C.len) \\ \equiv \quad & \text{false} \ \vee \ \text{true} \ \vee \ \text{true} \\ \equiv \quad & \text{true} \end{aligned}$$

$$\begin{aligned} & (G_{union}(i, j, k) \ \wedge \ implicit\_constraints(i, j, k))[i, j, k, r \backslash A.len, 0, 0, 0] \\ \equiv \quad & \\ & G_{union}(A.len, 0, 0) \ \wedge \ implicit\_constraints(A.len, 0, 0) \\ \equiv \quad & \\ & G_{union}(A.len, 0, 0) \ \wedge \ (A.len \in [0, A.len) \ \wedge \ 0 \in [0, B.len) \ \wedge \ 0 \in [0, C.len)) \\ \equiv \quad & \\ & G_{union}(A.len, 0, 0) \ \wedge \ (\text{false} \ \wedge \ \text{true} \ \wedge \ \text{true}) \\ \equiv \quad & \\ & G_{union}(A.len, 0, 0) \ \wedge \ \text{false} \\ \equiv \quad & \\ & \text{false} \end{aligned}$$

$\therefore$

$$\begin{aligned} (inv \ \wedge \ guard)[i, j, k, r \backslash A.len, 0, 0, 0] \quad & \not\Rightarrow \quad (G_{union}(i, j, k) \\ & \qquad \wedge \ implicit\_constraints(i, j, k))[i, j, k, r \backslash A.len, 0, 0, 0] \\ inv[i, j, k, r \backslash A.len, 0, 0, 0] \ \wedge \ guard[i, j, k, r \backslash A.len, 0, 0, 0] \quad & \not\Rightarrow \quad \text{false} \\ \text{true} \ \wedge \ \text{true} \quad & \not\Rightarrow \quad \text{false} \\ \text{true} \quad & \not\Rightarrow \quad \text{false} \end{aligned}$$

The specification does not refine to the provided program.

Instead of $(i \neq A.len \ \vee \ j \neq B.len \ \vee \ k \neq C.len)$ as the **do** guard, the choice of $(i \neq A.len \ \wedge \ j \neq B.len \ \wedge \ k \neq C.len)$ prevents out-of-bounds array indexing in the guards of the **if** statement, and could be refined to by the specification.

We can quickly show that this new guard, $guard'$, doesn't fall to the counter-example we used before,

$$
\begin{aligned}
guard'[i,j,k,r\backslash A.len,0,0,0] &\equiv (i \neq A.len \ \wedge \ j \neq B.len \ \wedge \ k \neq C.len)[i,j,k,r\backslash A.len,0,0,0] \\
&\equiv (A.len \neq A.len \ \wedge \ 0 \neq B.len \ \wedge \ 0 \neq C.len) \\
&\equiv \text{false} \ \wedge \ \text{true} \ \wedge \ \text{true} \\
&\equiv \text{false}
\end{aligned}
$$

s.t.

$$
\begin{aligned}
(inv \ \wedge \ guard')[i,j,k,r\backslash A.len,0,0,0] &\Rightarrow (G_{union}(i,j,k) \\
&\quad \wedge \ implicit\_constraints(i,j,k))[i,j,k,r\backslash A.len,0,0,0] \\
inv[i,j,k,r\backslash A.len,0,0,0] \ \wedge \ guard'[i,j,k,r\backslash A.len,0,0,0] &\Rightarrow \text{false} \\
\text{true} \ \wedge \ \text{false} &\Rightarrow \text{false} \\
\text{false} &\Rightarrow \text{false}
\end{aligned}
$$

and will terminate the loop before any out-of-bounds array indexing occurs.
Furthermore

$$
\begin{aligned}
inv \wedge guard' &\equiv D_{[0,r)} = A_{[0,i)} \cap B_{[0,j)} \cap C_{[0,k)} \\
&\quad \wedge \ r \in [0, D.len] \ \wedge \ i \in [0, A.len] \ \wedge \ j \in [0, B.len] \ \wedge \ k \in [0, C.len] \\
&\quad \wedge \ (i \neq A.len \ \wedge \ j \neq B.len \ \wedge \ k \neq C.len) \\
&\equiv D_{[0,r)} = A_{[0,i)} \cap B_{[0,j)} \cap C_{[0,k)} \\
&\quad \wedge \ r \in [0, D.len] \ \wedge \ i \in [0, A.len) \ \wedge \ j \in [0, B.len) \ \wedge \ k \in [0, C.len) \\
&\Rrightarrow G_{union}(i,j,k) \ \wedge \ implicit\_constraints(i,j,k)
\end{aligned}
$$