Assignment 2: Verification

Maxwell Bo 43926871

April 13, 2017

1 Part A

Let

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

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

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

where

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

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

$\because$

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

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

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

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

$\therefore$

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

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

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

$\because$

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

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

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

where

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

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

where

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

$\therefore$

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

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

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

With these new implicit constraints, we can conclude that

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

For convenience

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

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 $C.len = 1$.

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

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

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

$\therefore$

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

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 would 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.

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