

Assignment 2: Verification

Maxwell Bo 43926871

April 13, 2017

1 Part A

Let

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

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

$$i, j, k, r, D : [pre, post(r)]$$

$$\sqsubseteq \{ \text{Composition: middle predicate is } inv \}$$

$$i, j, k, r, D : [pre, inv]; i, j, k, r, D : [inv, post(r)]$$

where

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

$$\sqsubseteq \{ \text{Assignment: } pre \Rightarrow inv[i, j, k, r \setminus 0, 0, 0, 0] \}$$

$$i, j, k, r := 0, 0, 0, 0; i, j, k, r, D : [inv, post(r)]$$

\therefore

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

$$\sqsubseteq \{ \text{Strengthen post: } inv \wedge \neg guard \Rightarrow post(r) \}$$

$$i, j, k, r := 0, 0, 0, 0; i, j, k, r, D : [inv, inv \wedge \neg guard]$$

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

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

\therefore

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

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

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

where

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

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

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

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

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

With these new implicit constraints, we can conclude that

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

For convenience

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

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 $C.len = 1$.

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

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

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

\therefore

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

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,

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

s.t.

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

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

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