Programming Languages: Imperative Program Construction
Practicals 9: Array Manipulation

Shin-Cheng Mu

Autumn Term, 2024

Typical Array Manipulation

1. Given $a : \textbf{array}\ [0..10)\ \textbf{of}\ \mathit{Int}$, compute $\mathit{wp}\ (a[i] := 0)\ (a[2] \neq 0)$.

2. Given constant $N, Y : \mathit{Int}$ with $0 \leqslant N$, and variables $b : \textbf{array}\ [0..N)\ \textbf{of}\ \mathit{Int}$, $x, i : \mathit{Int}$,
   (a) compute $\mathit{wp}\ (b[i-1] := x+1)\ \langle \forall j : i \leqslant j < N : b[j] = Y \rangle$.
   (b) Compute $\mathit{wp}\ (b[i-1] := x+1; i := i-1)\ \langle \forall j : i \leqslant j < N : b[j] = Y \rangle$.

3. Derive

   $\textbf{con}\ N : \mathit{Int}\ \{1 \leqslant N\}$
   $\textbf{con}\ F : \textbf{array}\ [0..N)\ \textbf{of}\ \mathit{Int}$
   $\textbf{var}\ \ h : \textbf{array}\ [0..N)\ \textbf{of}\ \mathit{Int}$
   $\mathit{running\_sum}$
   $\{\langle \forall k : 0 \leqslant k < N : h[k] = \langle \Sigma i : 0 \leqslant i \leqslant k : F[i] \rangle \rangle\}$.

4. Derive

   $\textbf{con}\ N : \mathit{Int}\ \{1 \leqslant N\}$
   $\textbf{var}\ f\ : \textbf{array}\ [0..N)\ \textbf{of}\ \mathit{Int}$
   $\textbf{con}\ H : \textbf{array}\ [0..N)\ \textbf{of}\ \mathit{Int}$
   $\mathit{decompose}$
   $\{\langle \forall k : 0 \leqslant k < N : H[k] = \langle \Sigma i : 0 \leqslant i \leqslant k : f[i] \rangle \rangle\}$.

Swaps

5. Prove that

   $\{h[0] = 0 \wedge h[1] = 1\}$ -- hence $h[h[0]] = 0$
   $\mathit{swap}\ h\ (h[0])\ (h[1])$
   $\{h[h[1]] = 1\}$

6. Given $h : \textbf{array}\ [0..N)\ \textbf{of}\ A$, prove the rule that when h does not occur free in E and F,

   $\{\langle \forall i : 0 \leqslant i < N \wedge i \neq E \wedge i \neq F : h[i] = H\ i \rangle \wedge h[E] = X \wedge h[F] = Y\}$
   $\mathit{swap}\ h\ E\ F$
   $\{\langle \forall i : 0 \leqslant i < N \wedge i \neq E \wedge i \neq F : h[i] = H\ i \rangle \wedge h[E] = Y \wedge h[F] = X\}$.

   Notes:

- Recall that E and F are expressions, while X, Y, H are logical variables. It means that, for example, one can conclude immediately $X[z\backslash w] = X$ for $z \not\equiv X$, while to determine whether $E[z\backslash w] = E$ we have to look into E — $E[z\backslash w] = E$ if z does not occur free in E.
- With $h[E] = X$, for example, we implicitly assume that *def* $(h[E])$ holds.

7. Derive the following program, where arrays are manipulated only by swapping.

   con $N : Int$ $\{0 \leqslant N\}$
   var h : **array** $[0..N)$ **of** Int
   var $p : Int$
   ?
   $\{0 \leqslant p \leqslant N \wedge \langle \forall i : 0 \leqslant i < p : h[i] \leqslant 0 \rangle \wedge \langle \forall i : p \leqslant i < N : 0 \leqslant h[i] \rangle\}$.

8. The following is a specification of sorting:

   con $N : Int$ $\{0 \leqslant N\}$
   var h : **array** $[0..N)$ **of** Int
   sort
   $\{\langle \forall i\ j : 0 \leqslant i \leqslant j < N : h[i] \leqslant h[j] \rangle\}$.

   where *sort* mutates the array h only by swapping. Derive a $O(N^2)$ algorithm for sorting. The algorithm will contain a loop within a loop. The outer loop uses as invariant $P_0 \wedge P_1$, where

   $P_0 \equiv \langle \forall i : 0 \leqslant i < n : \langle \forall j : i \leqslant j < N : h[i] \leqslant h[j] \rangle \rangle$,
   $P_1 \equiv 0 \leqslant n \leqslant N$.

   The inner loop uses Q as *part of* its invariant:

   $Q \equiv \langle \forall j : k \leqslant j < N : h[n] \leqslant h[j] \rangle$.