Programming Languages: Imperative Program Construction 9. Array Manipulation

Shin-Cheng Mu

Autumn Term, 2021

National Taiwan University and Academia Sinica

Some Notes on Definedness

- Recall the weakest precondition for assignments:

$$wp\ (x := E)\ P = P[x \backslash E] \quad .$$

- That is not the whole story... since we have to be sure that E is defined!

- In our current language, given expression E there is a systematic (inductive) definition on what needs to be proved to ensure that E is defined. Let's denote it by $\mathit{def}\ E$.
- We will not go into the detail but give examples.
- For example, if there is division in E, the denominator must not be zero.
    - $\mathit{def}\,(x + y \,/\, (z + x)) = (z + x \neq 0)$.
    - $\mathit{def}\,(x + y \,/\, 2) = (2 \neq 0) = \mathit{True}$.

- A more complete rule:

$$wp\ (x := E)\ P = P[x \backslash E] \land \mathit{def}\ E \quad .$$

- In fact, all expressions need to be defined. E.g.

$$\begin{aligned} &wp\ (\textbf{if}\ B_0 \to S_0 \mid B_1 \to S_1)\ P = \\ &\quad B_0 \Rightarrow wp\ S_0\ P \land B_1 \Rightarrow wp\ S_1\ P \land (B_0 \lor B_1) \land \\ &\quad \mathit{def}\ B_0 \land \mathit{def}\ B_1 \quad . \end{aligned}$$

- How come we have never mentioned so?
- The first partial operation we have used was division. And the denominator was usually a constant (namely, 2!).

Array Bound

- Array indexing is a partial operation too — we need to be sure that the index is within the domain of the array.
- Let $A : \textbf{array } [M..N) \textbf{ of } \mathit{Int}$ and let I be an expression. We define $\mathit{def}\ (A[I]) = \mathit{def}\ I \wedge M \leqslant I < N$.
- E.g. given $A : \textbf{array } [0..N) \textbf{ of } \mathit{Int}$,
  - $\mathit{def}\ (A[x\ /\ z] + A[y]) = z \neq 0 \wedge 0 \leqslant x\ /\ z < N \wedge 0 \leqslant y < N$.
  - $\mathit{wp}\ (s := s \uparrow A[n])\ P = P[s \backslash s \uparrow A[n]] \wedge 0 \leqslant n < N$.
- We never made it explicit, because conditions such as $0 \leqslant n < N$ were usually already in the invariant/guard and thus discharged immediately.

Array Assignment

- So far, all our arrays have been constants — we read from the arrays but never wrote to them!
- Consider $a : \textbf{array } [0..2) \textbf{ of } \mathit{Int}$, where $a[0] = 0$ and $a[1] = 1$.
- It should be true that

$$\begin{aligned}&\{a[0] = 0 \land a[1] = 1\}\\ &a[a[1]] := 0\\ &\{a[a[1]] = 1\} \quad .\end{aligned}$$

- However, if we use the previous *wp*,

$$\begin{aligned}&\quad \mathit{wp}\ (a[a[1]] := 0)\ (a[a[1]] = 1)\\ &\equiv (a[a[1]] = 1)[a[a[1]]\backslash 0]\\ &\equiv 0 = 1\\ &\equiv \mathit{False} \quad .\end{aligned}$$

- What went wrong?

- For a more obvious example where our previous *wp* does not work for array assignment:
- $wp\ (a[i] := 0)\ (a[2] \neq 0)$ appears to be $a[2] \neq 0$, since $a[i]$ does not appear (verbatim) in $a[2] \neq 0$.
- But what if $i = 2$?

- An array is a function. E.g. $a : \textbf{array } [0..N) \textbf{ of } \mathit{Bool}$ is a function $\mathit{Int} \to \mathit{Bool}$ whose domain is $[0..N)$.
- Indexing $a[n]$ is function application.
    - Some textbooks use the same notation for function application and array indexing.
    - (Could that have been a better choice for this course?)

- Given $f : A \to B$, let $(f{:}x \mapsto e)$ denote the function that *maps* x *to* e, and otherwise the same as f.

$$
\begin{aligned}
(f{:}x \mapsto e)\ y &= e &&, \text{if } x = y;\\
&= f\ y &&, \text{otherwise.}
\end{aligned}
$$

- For example, given $f\,x = x^2$, $(f{:}1 \mapsto -1)$ is a function such that

$$
\begin{aligned}
(f{:}1 \mapsto -1)\ 1 &= -1 &&,\\
(f{:}1 \mapsto -1)\ x &= x^2 &&, \text{if } x \neq -1.
\end{aligned}
$$

- Key: assignment to array should be understood as altering the entire function.
- Given $a : \textbf{array}\ [M..N)\ \textbf{of}\ A$ (for any type A), the updated rule:

$$wp\ (a[I] := E)\ P = P[a \backslash (a : I \mapsto E)] \wedge \\ def\ (a[I]) \wedge def\ E \quad .$$

- In our examples, $def\ (a[I])$ and $def\ E$ can often be discharged immediately. For example, the boundary check $M \leqslant I < N$ can often be discharged soon. But do not forget about them.

- Recall our example

$$\begin{aligned} &\{a[0] = 0 \land a[1] = 1\} \\ &a[a[1]] := 0 \\ &\{a[a[1]] = 1\} \quad . \end{aligned}$$

- We aim to prove

$$\begin{aligned} &a[0] = 0 \land a[1] = 1 \Rightarrow \\ &\quad wp\ (a[a[1]] := 0)\ (a[a[1]] = 1) \quad . \end{aligned}$$

Assume $a[0] = 0 \land a[1] = 1$.

$$
\begin{aligned}
& \quad wp\ (a[a[1]] := 0)\ (a[a[1]] = 1) \\
\equiv & \quad \{ \text{ def. of } wp \text{ for array assignment } \} \\
& \quad (a : a[1] \mapsto 0)[(a : a[1] \mapsto 0)[1]] = 0 \\
\equiv & \quad \{ \text{ assumption: } a[1] = 1 \ \} \\
& \quad (a : 1 \mapsto 0)[(a : 1 \mapsto 0)[1]] = 0 \\
\equiv & \quad \{ \text{ def. of alteration: } (a : 1 \mapsto 0)[0] = 0 \ \} \\
& \quad (a : 1 \mapsto 0)[0] = 0 \\
\equiv & \quad \{ \text{ def. of alteration: } (a : 1 \mapsto 0)[0] = a[0] \ \} \\
& \quad a[0] = 0 \\
\equiv & \quad \{ \text{ assumption: } a[0] = 0 \ \} \\
& \quad \mathit{True} \quad .
\end{aligned}
$$

- In this course, parallel assignments to arrays are not allowed.
- This is done to avoid having to define what the following program ought to do:

$$
\begin{aligned}
&x, y := 0, 0; \\
&a[x], a[y] := 0, 1
\end{aligned}
$$

- It is possible to give such programs a definition (e.g. choose an order), but we prefer to keep it simple.

Typical Array Manipulation in a Loop

Consider:

$$\begin{aligned}
&\textbf{con}\ N : \mathit{Int}\ \{0 \leqslant N\}\\
&\textbf{var}\ h : \textbf{array}\ [0..N)\ \textbf{of}\ \mathit{Int}\\
&\mathit{allzeros}\\
&\{\langle \forall i : 0 \leqslant i < N : h[i] = 0 \rangle\}
\end{aligned}$$

$$
\begin{array}{l}
\textbf{con}\ N : Int\ \{0 \leqslant N\} \\
\textbf{var}\ h : \textbf{array}\ [0..N)\ \textbf{of}\ Int \\
\textbf{var}\ n : Int \\
n := 0 \\
\{\langle \forall i : 0 \leqslant i < n : h[i] = 0 \rangle \wedge 0 \leqslant n \leqslant N, \\
\quad bnd : N - n\} \\
\textbf{do}\ n \neq N \rightarrow\ ? \\
\qquad\qquad\qquad n := n + 1 \\
\textbf{od} \\
\{\langle \forall i : 0 \leqslant i < N : h[i] = 0 \rangle\}
\end{array}
$$

- With $0 \leqslant n \leqslant N \wedge n \neq N$:

$$
\begin{aligned}
& \langle \forall i : 0 \leqslant i < n : h[i] = 0 \rangle [n \backslash n + 1] \\
\equiv\ & \langle \forall i : 0 \leqslant i < n + 1 : h[i] = 0 \rangle \\
\equiv\ & \quad \{ \text{ split off } i = n \ \} \\
& \langle \forall i : 0 \leqslant i < n : h[i] = 0 \rangle \wedge h[n] = 0 \quad .
\end{aligned}
$$

- If we conjecture that ? is an assignment $h[I] := E$, we ought to find I and E such that the following can be satisfied:

$$
\begin{aligned}
& \langle \forall i : 0 \leqslant i < n : h[i] = 0 \rangle \wedge 0 \leqslant n < N \Rightarrow \\
& \quad \langle \forall i : 0 \leqslant i < n : (h : I \mapsto E)[i] = 0 \rangle \wedge \\
& \quad (h : I \mapsto E)[n] = 0 \quad .
\end{aligned}
$$

- An obvious choice: $(h : n \mapsto 0)$,
- which almost immediately leads to

$$
\begin{array}{ll}
 & \langle \forall i : 0 \leqslant i < n : (h : n \mapsto 0)[i] = 0 \rangle \wedge \\
 & (h : n \mapsto 0)[n] = 0 \\
\equiv & \quad \{ \text{ function alteration } \} \\
 & \langle \forall i : 0 \leqslant i < n : h[i] = 0 \rangle \wedge 0 = 0 \\
\Leftarrow & \langle \forall i : 0 \leqslant i < n : h[i] = 0 \rangle \wedge 0 \leqslant n < N \quad .
\end{array}
$$

$$
\begin{array}{l}
\textbf{con}\ N : \mathit{Int}\ \{0 \leqslant N\} \\
\textbf{var}\ h : \textbf{array}\ [0..N)\ \textbf{of}\ \mathit{Int} \\
\textbf{var}\ n : \mathit{Int} \\
n := 0 \\
\{\langle \forall i : 0 \leqslant i < n : h[i] = 0 \rangle \wedge 0 \leqslant n \leqslant N, \\
\quad \mathit{bnd} : N - n\} \\
\textbf{do}\ n \neq N \rightarrow h[n] := 0; n := n + 1\ \textbf{od} \\
\{\langle \forall i : 0 \leqslant i < N : h[i] = 0 \rangle\}
\end{array}
$$

Obvious, but useful.

- The calculation can certainly be generalised.
- Given a function $H : Int \to A$, and suppose we want to establish

$$\langle \forall i : 0 \leqslant i < N : h[i] = H\ i \rangle \quad ,$$

where H *does not depend on* h (e.g, h does not occur free in H).

- Let $P\ n = 0 \leqslant n < N \wedge \langle \forall i : 0 \leqslant i < n : h[i] = H\ i \rangle)$.
- We aim to establish $P\ (n+1)$, given $P\ n \wedge n \setminus = N$.

- One can prove the following:

$$\{P\ n \wedge n \setminus = N \wedge E = H\ n\}$$
$$h[n] := E$$
$$\{P\ (n+1)\}\quad ,$$

- which can be used in a program fragment...

$$
\begin{array}{l}
\{P\ 0\} \\
n := 0 \\
\{P\ n, bnd : N - n\} \\
\mathbf{do}\ n \neq N \rightarrow \\
\qquad \{\ \text{establish}\ E = H\ n\ \} \\
\quad h[n] := E \\
\quad n := n + 1 \\
\mathbf{od} \\
\{\langle \forall i : 0 \leqslant i < N : h[i] = H\ i \rangle\}
\end{array}
$$

- Why do we need E? Isn't E simply $H\ n$?
- In some cases $H\ n$ can be computed in one expression. In such cases we can simply do $h[n] := H\ n$.
- In some cases E may refer to previously computed results — other variables, or even h.
    - Yes, E may refer to h while H does not. There are such examples in the Practicals.

Consider:

$$\begin{array}{l}
\textbf{con}\ N : \mathit{Int}\ \{0 \leqslant N\}; X : \textbf{array}\ [0..N)\ \textbf{of}\ \mathit{Int} \\
\{\langle \forall i : 0 \leqslant i < N : 1 \leqslant X[i] \leqslant 6 \rangle\} \\
\textbf{var}\ h : \textbf{array}\ [1..6]\ \textbf{of}\ \mathit{Int} \\
\mathit{histogram} \\
\{\langle \forall i : 0 \leqslant i \leqslant 6 : h[i] = \\
\quad \langle \# k : 0 \leqslant k < N : X[k] = i \rangle \rangle\}
\end{array}$$

- Let $P\ n$ denote
  $\langle \forall i : 0 \leqslant i \leqslant 6 : h[i] = \langle \# k : 0 \leqslant k < n : X[k] = i \rangle \rangle$.
- A program skeleton:

$$
\begin{array}{l}
\textbf{con } N : \mathit{Int}\ \{0 \leqslant N\}; X : \textbf{array } [0..N) \textbf{ of } \mathit{Int} \\
\{\langle \forall i : 0 \leqslant i < N : 1 \leqslant X[i] \leqslant 6 \rangle\} \\
\textbf{var } h : \textbf{array } [1..6] \textbf{ of } \mathit{Int}; n : \mathit{Int} \\
\mathit{initialise} \\
n := 0 \\
\{P\ n \wedge 0 \leqslant n \leqslant N, \mathit{bnd} : N - n\} \\
\textbf{do } n \neq N \rightarrow\ ? \\
\qquad\qquad\qquad n := n + 1 \\
\textbf{od} \\
\{\langle \forall i : 0 \leqslant i \leqslant 6 : h[i] = \\
\quad \langle \# k : 0 \leqslant k < N : X[k] = i \rangle \rangle\}
\end{array}
$$

- The *initialise* fragment has to satisfy $P\ 0$, that is

$$
\begin{aligned}
& \langle \forall i : 0 \leqslant i \leqslant 6 : h[i] = \langle \# k : 0 \leqslant k < 0 : X[k] = i \rangle \rangle \\
\equiv\ & \langle \forall i : 0 \leqslant i \leqslant 6 : h[i] = 0 \rangle \quad ,
\end{aligned}
$$

- which can be performed by *allzeros*.

- Let's calculate $P\ (n + 1)$, assuming $0 \leqslant n < N$:

$$
\begin{aligned}
& \langle \forall i : 0 \leqslant i \leqslant 6 : h[i] = \\
& \quad \langle \# k : 0 \leqslant k < n + 1 : X[k] = i \rangle \rangle \\
\equiv & \qquad \{ \text{ split off } k = n \ \} \\
& \langle \forall i : 0 \leqslant i \leqslant 6 : h[i] = \\
& \quad \langle \# k : 0 \leqslant k < n : X[k] = i \rangle + \#(X[n] = i) \rangle
\end{aligned}
$$

- Recall that $\# : \mathit{Bool} \to \mathit{Int}$ is the function such that

$$
\begin{aligned}
\#\ \mathit{False} &= 0 \\
\#\ \mathit{True} &= 1 \quad .
\end{aligned}
$$

- Again we conjecture that $h[I] := E$ will do the trick.
- We want to find I ane E such that
  $P\ n \wedge 0 \leqslant n < N \Rightarrow (P\ (n+1))[h \backslash (h : I \mapsto E)]$ can be proved.
- Assume $P\ n \wedge 0 \leqslant n < N$, consider $(P\ (n+1))[h \backslash (h : I \mapsto E)]$

$$
\begin{array}{ll}
& \langle \forall i : 0 \leqslant i \leqslant 6 : (h : I \mapsto E)[i] = \\
& \quad \langle \# k : 0 \leqslant k < n : X[k] = i \rangle + \#(X[n] = i) \rangle \\
\equiv & \quad \{ P\ n \} \\
& \langle \forall i : 0 \leqslant i \leqslant 6 : (h : I \mapsto E)[i] = \\
& \quad h[i] + \#(X[n] = i) \rangle \\
\equiv & \quad \{ \text{defn. of } \# \} \\
& \langle \forall i : 0 \leqslant i \leqslant 6 : (h : I \mapsto E)[i] = V\ i \rangle, \textbf{where} \\
& \quad V\ i = h[i] + 1 \ \ , \text{if } X[n] = i; \\
& \quad \qquad\quad h[i] \ \ , \text{if } X[n] \neq i. \\
\equiv & \quad \{ \text{function alteration} \} \\
& \langle \forall i : 0 \leqslant i \leqslant 6 : (h : I \mapsto E)[i] = \\
& \quad (h : X[n] \mapsto h[i] + 1)[i] \rangle \quad .
\end{array}
$$

- Therefore one chooses $I = X[n]$ and $E = h[X[n]] + 1$.

The Program

Let $P\ n \equiv \langle \forall i : 0 \leqslant i \leqslant 6 : h[i] = \langle \# k : 0 \leqslant k < n : X[k] = i \rangle \rangle$.

$\mathbf{con}\ N : Int\ \{0 \leqslant N\}; X : \mathbf{array}\ [0..N)\ \mathbf{of}\ Int$

$\{\langle \forall i : 0 \leqslant i < N : 1 \leqslant X[i] \leqslant 6 \rangle\}$

$\mathbf{var}\ h : \mathbf{array}\ [1..6]\ \mathbf{of}\ Int$

$\mathbf{var}\ n : Int$

$n := 1$

$\mathbf{do}\ n \neq 7 \to h[n] := 0; n := n + 1\ \mathbf{od}$

$\{P\ 0\}$

$n := 0$

$\{P\ n \wedge 0 \leqslant n \leqslant N, bnd : N - n\}$

$\mathbf{do}\ n \neq N \to h[X[n]] := h[X[n]] + 1$

$\qquad\qquad n := n + 1$

$\mathbf{od}$

$\{\langle \forall i : 0 \leqslant i \leqslant 6 : h[i] =$

$\quad \langle \# k : 0 \leqslant k < N : X[k] = i \rangle \rangle\}$

SWAPS

- Given array $h\ [0..N)$ and integer expressions E and F, we abbreviate the code fragment

$$|[\ \mathbf{var}\ r;\ r := h[E];\ h[E] := h[F];\ h[F] := r\ ]|$$

to *swap h E F*.
- $|[...]|$ denotes a program block with local constants and variables. We have not used this feature so far.
- Intuitively, *swap h E F* means “swapping the values of $h[E]$ and $h[F]$. (See the notes below, however.)

- We also extend the notion of function alteration to two entries.

$$
\begin{aligned}
(f{:}x, y \mapsto e, f)\; z &= e \quad , \text{if } z = x, \\
&= f \quad , \text{if } z = y, \\
&= f\, z \;\; , \text{otherwise.}
\end{aligned}
$$

- We have

$$
\begin{aligned}
& wp\ (swap\ h\ E\ F)\ P = def\ (h[E]) \wedge def\ (h[F]) \wedge \\
& \quad P[h \backslash (h : E, F \mapsto h[F], h[E])] \quad .
\end{aligned}
$$

- Note that it is *not* always the case that

$$\{h[E] = X\}\ \mathit{swap}\ h\ E\ F\ \{h[F] = X\} \quad .$$

- Consider $h[0] = 0 \wedge h[1] = 1$. This does not hold:

$$\{h[h[0]] = 0\}\ \mathit{swap}\ h\ (h[0])\ (h[1])\ \{h[h[1]] = 0\} \quad .$$

- In fact, after swapping we have $h[0] = 1 \wedge h[1] = 0$, and hence $h[h[1]] = 1$.

A Simpler Case

- However, when h does not occur free in E and F, we do have

$$
\begin{aligned}
&(\{\langle\forall i : i \neq E \wedge i \neq F : h[i] = H\ i\rangle\} \wedge \\
&\quad h[E] = X \wedge h[E] = Y) \\
&\mathit{swap}\ h\ E\ F \\
&(\{\langle\forall i : i \neq E \wedge i \neq F : h[i] = H\ i\rangle\} \wedge \\
&\quad h[E] = Y \wedge h[E] = X) \quad .
\end{aligned}
$$

- It is a convenient rule we use when reasoning about swapping.

The Dutch National Flag

- Let $RWB = \{R, W, B\}$ (standing respecively for red, white, and blue).

$$
\begin{array}{l}
\textbf{con}\ N : Int\ \{0 \leqslant N\} \\
\textbf{var}\ h : \textbf{array}\ [0..N)\ \textbf{of}\ RWB \\
\textbf{var}\ r, w : Int \\
dutch\_national\_flag \\
\{0 \leqslant r \leqslant w \leqslant N \wedge \\
\quad \langle \forall i : 0 \leqslant i < r : h[i] = R \rangle \wedge \\
\quad \langle \forall i : r \leqslant i < w : h[i] = W \rangle \wedge \\
\quad \langle \forall i : w \leqslant i < N : h[i] = B \rangle \wedge \}
\end{array}
$$

- The program shall manipulate h only by swapping.
- Denote the postcondition by Q.

- Introduce a variable b.
- Choose as invariant $P_0 \wedge P_1$, where

$$
\begin{aligned}
P_0 &\equiv P_r \wedge P_w \wedge P_b \\
P_1 &\equiv 0 \leqslant r \leqslant w \leqslant b \leqslant N \\
P_r &\equiv \langle \forall i : 0 \leqslant i < r : \; h[i] = R \rangle \\
P_w &\equiv \langle \forall i : r \leqslant i < w : \; h[i] = W \rangle \\
P_b &\equiv \langle \forall i : b \leqslant i < N : h[i] = B \rangle
\end{aligned}
$$

- $P_0 \wedge P_1$ can be established by $r, w, b := 0, 0, N$.
- If $w = b$, we get the postcondition Q.

The Plan

$$
\begin{array}{l}
r, w, b := 0, 0, N \\
\{P_0 \wedge P_1, bnd : b - w\} \\
\textbf{do } b \neq w \rightarrow \textbf{if } h[w] = R \rightarrow S_r \\
\qquad\qquad\qquad \mid h[w] = W \rightarrow S_w \\
\qquad\qquad\qquad \mid h[w] = B \rightarrow S_b \\
\qquad\qquad\quad \textbf{fi} \\
\textbf{od} \\
\{Q\}
\end{array}
$$

Observation

- Note that
  - r is the number of red elements detected,
  - $w - r$ is the number of white elements detected,
  - $N - b$ is the number of blue elements detected.
- Therefore, S_w should contain $w := w + 1$, S_b should contain $b := b - 1$·
- S_r should contain $r, w := r + 1, w + 1$, thus r increases but $w - r$ is unchanged.
- The bound decreases in all cases! Good sign.

- The case for white is the easiest, since

$$P_0 \wedge P_1 \wedge h[w] = W \Rightarrow (P_0 \wedge P_1)[w \backslash w + 1] \ .$$

- It is sufficient to let S_w be simply $w := w + 1$.

- We have

$$\{P_r \wedge P_w \wedge P_b \wedge w < b \wedge h[w] = B\}$$
$$swap\ h\ w\ (b - 1)$$
$$\{P_r \wedge P_w \wedge P_b \wedge w < b \wedge h[b - 1] = B\}$$
$$b := b - 1$$
$$\{P_r \wedge P_w \wedge P_b \wedge w \leqslant b\}$$

- Thus we choose $swap\ h\ w\ (b - 1); b := b - 1$ as S_b.

- Precondition: $P_r \wedge P_w \wedge P_b \wedge w < b \wedge h[w] = R$.
- It appears that *swap h w r* establishes $P[w \backslash w + 1]$. But we have to see what $h[r]$ is before we can increment r.
- P_w implies $r < w \Rightarrow h[r] = W$. Equivalently, we have $r = w \vee h[r] = W$.

Red: Case $r = w$

- We have

$$
\begin{aligned}
&\{P_r \wedge P_w \wedge P_b \wedge r = w < b \wedge h[w] = R\} \\
&\mathit{swap}\ h\ w\ r \\
&\{P_r \wedge P_w \wedge P_b \wedge w < b \wedge h[r] = R\} \\
&r, w := r + 1, w + 1 \\
&\{P_r \wedge P_w \wedge P_b \wedge r = w \leqslant b\}
\end{aligned}
$$

RED: CASE $h[r] = W$

- We have

$$\{P_r \wedge P_w \wedge P_b \wedge w < b \wedge h[r] = W \,\&\, h[w] = R\}$$
$$swap\ h\ w\ r$$
$$\{P_r \wedge h[r] = R \wedge \langle \forall i : r + 1 \leqslant i < w : h[i] = W \rangle \wedge$$
$$\quad P_b \wedge w < b\}$$
$$r, w := r + 1, w + 1$$
$$\{P_r \wedge P_w \wedge P_b \wedge r = w \leqslant b\}$$

- In both cases, $swap\ h\ w\ r; r, w := r + 1, w + 1$ is a valid choice.

$$
\begin{array}{l}
\textbf{con } N : Int \; \{0 \leqslant N\} \\
\textbf{var } h : \textbf{array } [0..N) \textbf{ of } RWB \\
\textbf{var } r, w, b : Int \\
r, w, b := 0, 0, N \\
\{P_0 \wedge P_1, bnd : b - w\} \\
\textbf{do } b \neq w \rightarrow \textbf{if } \begin{array}[t]{lll}
h[w] = R & \rightarrow & swap\ h\ w\ r \\
 & & r, w := r + 1, w + 1 \\
\mid h[w] = W & \rightarrow & w := w + 1 \\
\mid h[w] = B & \rightarrow & swap\ h\ w\ (b - 1) \\
 & & b := b - 1 \\
\textbf{fi} & &
\end{array} \\
\textbf{od} \\
\{Q\}
\end{array}
$$