Programming Languages: Imperative Program Construction
9. Array Manipulation

Shin-Cheng Mu

Autumn Term, 2021

National Taiwan University and Academia Sinica

Materials in these notes are mainly from Kaldewaij. Some examples are adapted from the course CSci 550: Program Semantics and Derivation taught by Prof. H. Conrad Cunningham, University of Mississippi.

Some Notes on Definedness

- Recall the weakest precondition for assignments:

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

- 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 *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.
  - *def* $(x + y / (z + x)) = (z + x \neq 0)$.
  - *def* $(x + y / 2) = (2 \neq 0) = \textit{True}$.

- A more complete rule:

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

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

  $$wp\ (\textbf{if}\ B_0 \to S_0 \mid B_1 \to S_1\ \textbf{fi})\ P =$$
  $$B_0 \Rightarrow wp\ S_0\ P \land B_1 \Rightarrow wp\ S_1\ P \land (B_0 \lor B_1) \land$$
  $$def\ B_0 \land def\ B_1\ .$$

- 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 indexing is a partial operation too — we need to be sure that the index is within the domain of the array.
- Let $A :$ **array** $[M..N)$ **of** Int and let I be an expression. We define $def\ (A[I]) = def\ I \wedge M \leqslant I < N$.
- E.g. given $A :$ **array** $[0..N)$ **of** Int,
    - $def\ (A[x\ /\ z] + A[y]) = z \neq 0 \wedge 0 \leqslant x\ /\ z < N \wedge 0 \leqslant y < N$.
    - $wp\ (s := s \uparrow A[n])\ P = P[s \backslash s \uparrow A[n]] \wedge 0 \leqslant n < N$.
- 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

ARRAY ASSIGNMENT

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

    $$\{a[0] = 1 \land a[1] = 1\}$$
    $$a[a[1]] := 0$$
    $$\{a[a[1]] = 1\}\ .$$

- However, if we use the previous wp,

    $$wp\ (a[a[1]] := 0)\ (a[a[1]] = 1)$$
    $$\equiv (a[a[1]] = 1)[a[a[1]]\backslash 0]$$
    $$\equiv 0 = 1$$
    $$\equiv False\ .$$

- 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 : **array** [0..*N*) **of** *Bool* is a function *Int* → *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*.

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

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

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

- Key: assignment to array should be understood as altering the entire function.

- Given $a :$ **array** $[M..N)$ **of** A (for any type A), the updated rule:

$$wp\ (a[I] := E)\ P = P[a \setminus (a : I \rightarrow E)]\ \wedge$$
$$def\ (a[I]) \wedge def\ E\ \ .$$

- 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

$$\{a[0] = 1 \land a[1] = 1\}$$
$$a[a[1]] := 0$$
$$\{a[a[1]] = 1\} \ .$$

- We aim to prove

$$a[0] = 1 \land a[1] = 1 \Rightarrow$$
$$wp \ (a[a[1]] := 0) \ (a[a[1]] = 1) \ .$$

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

$$wp \ (a[a[1]] := 0) \ (a[a[1]] = 1)$$
$$\equiv \quad \{ \text{ def. of } wp \text{ for array assignment } \}$$
$$(a : a[1] \rightarrow 0)[(a : a[1] \rightarrow 0)[1]] = 1$$
$$\equiv \quad \{ \text{ assumption: } a[1] = 1 \}$$
$$(a : 1 \rightarrow 0)[(a : 1 \rightarrow 0)[1]] = 1$$
$$\equiv \quad \{ \text{ def. of alteration: } (a : 1 \rightarrow 0)[0] = 0 \}$$
$$(a : 1 \rightarrow 0)[0] = 1$$
$$\equiv \quad \{ \text{ def. of alteration: } (a : 1 \rightarrow 0)[0] = a[0] \}$$
$$a[0] = 1$$
$$\equiv \quad \{ \text{ assumption: } a[0] = 1 \}$$
$$True \ .$$

- 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:

  $x, y := 0, 0;$
  $a[x], a[y] := 0, 1$

- 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:

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

con $N : Int$ $\{0 \leqslant N\}$
var $h :$ array $[0..N)$ of Int
var $n : Int$

$n := 0$
$\{\langle \forall i : 0 \leqslant i < n : h[i] = 0 \rangle \wedge 0 \leqslant n \leqslant N,$
  $bnd : N - n\}$
do $n \neq N \rightarrow$?
          $n := n + 1$
od
$\{\langle \forall i : 0 \leqslant i < N : h[i] = 0 \rangle\}$

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

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

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

$$\langle \forall i : 0 \leqslant i < n : h[i] = 0 \rangle \wedge 0 \leqslant n < N \Rightarrow$$
$$\langle \forall i : 0 \leqslant i < n : (h : I \rightarrow E)[i] = 0 \rangle \wedge$$
$$(h : I \rightarrow E)[n] = 0 \ .$$

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

$$\langle \forall i : 0 \leqslant i < n : (h : n \shortrightarrow 0)[i] = 0 \rangle \,\wedge$$
$$(h : n \shortrightarrow 0)[n] = 0$$
$$\equiv \quad \{ \text{ function alteration } \}$$
$$\langle \forall i : 0 \leqslant i < n : h[i] = 0 \rangle \wedge 0 = 0$$
$$\Leftarrow \langle \forall i : 0 \leqslant i < n : h[i] = 0 \rangle \wedge 0 \leqslant n < N \quad.$$

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

Obvious, but useful.

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

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

  where H *does not depend on* h (e.g, h does not occur free in H).
- Let $P\ n = 0 \leqslant n < N \land \langle \forall i : 0 \leqslant i < n : h[i] = H\ i \rangle)$.
- We aim to establish $P\ (n + 1)$, given $P\ n \land n \neq N$.

- One can prove the following:

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

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

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

- 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:

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

- 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:

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

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

$$\langle \forall i : 1 \leqslant i \leqslant 6 : h[i] = \langle \#k : 0 \leqslant k < 0 : X[k] = i \rangle \rangle$$
$$\equiv \langle \forall i : 1 \leqslant i \leqslant 6 : h[i] = 0 \rangle \;,$$

- which can be performed by *allzeros*.

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

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

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

$$\# \textit{ False} = 0$$
$$\# \textit{ True } = 1 \ .$$

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

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

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

The Program

Let $P\,n \equiv \langle \forall i : 1 \leqslant i \leqslant 6 : h[i] = \langle \#k : 0 \leqslant k < n : X[k] = i \rangle \rangle$.

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

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

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

$\textbf{var } n : \textit{Int}$

$n := 1$

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

$\{P\,0\}$

$n := 0$

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

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

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

$\textbf{od}$

$\{\langle \forall i : 1 \leqslant i \leqslant 6 : h[i] =$