

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!

DEFINEDNESS

- 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 $\text{def } E$.
- We will not go into the detail but give examples.
- For example, if there is division in E , the denominator must not be zero.
 - $\text{def } (x + y / (z + x)) = (z + x \neq 0)$.
 - $\text{def } (x + y / 2) = (2 \neq 0) = \text{True}$.

- A more complete rule:

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

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

$$\begin{aligned} wp\ (\text{if } B_0 \rightarrow S_0 \mid B_1 \rightarrow S_1\ \text{fi})\ P = \\ B_0 \Rightarrow wp\ S_0\ P \wedge B_1 \Rightarrow wp\ S_1\ P \wedge (B_0 \vee B_1) \wedge \\ def\ B_0 \wedge def\ B_1\ . \end{aligned}$$

HOW COME WE HAVE NEVER MENTIONED SO?

- 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 $A : \text{array } [M..N) \text{ of } Int$ and let I be an expression. We define $def(A[I]) = def I \wedge M \leq I < N$.
- E.g. given $A : \text{array } [0..N) \text{ of } Int$,
 - $def(A[x / z] + A[y]) = z \neq 0 \wedge 0 \leq x / z < N \wedge 0 \leq y < N$.
 - $wp(s := s \uparrow A[n]) P = P[s \setminus s \uparrow A[n]] \wedge 0 \leq n < N$.
- We never made it explicit, because conditions such as $0 \leq n < N$ 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 $a : \text{array } [0..2) \text{ of } \text{Int}$, where $a[0] = 1$ and $a[1] = 1$.
- It should be true that

$$\{a[0] = 1 \wedge a[1] = 1\}$$

$$a[a[1]] := 0$$

$$\{a[a[1]] = 1\} .$$

- However, if we use the previous wp ,

$$wp(a[a[1]] := 0) (a[a[1]] = 1)$$

$$\equiv (a[a[1]] = 1)[a[a[1]] \setminus 0]$$

$$\equiv 0 = 1$$

$$\equiv \text{False} .$$

- What went wrong?

ANOTHER COUNTEREXAMPLE

- 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 $i = 2$?

ARRAYS AS FUNCTIONS

- An array is a function. E.g. $a : \text{array } [0..N) \text{ of } \text{Bool}$ is a function $\text{Int} \rightarrow \text{Bool}$ 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?)

FUNCTION ALTERATION

- Given $f: A \rightarrow B$, 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 $a : \text{array } [M..N] \text{ of } A$ (for any type A), the updated rule:

$$\text{wp } (a[l] := E) P = P[a \setminus (a : l \mapsto E)] \wedge \\ \text{def } (a[l]) \wedge \text{def } E .$$

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

THE EXAMPLE

- Recall our example

$$\{a[0] = 1 \wedge a[1] = 1\}$$

$$a[a[1]] := 0$$

$$\{a[a[1]] = 1\} .$$

- We aim to prove

$$a[0] = 1 \wedge a[1] = 1 \Rightarrow$$

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

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

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

RESTRICTIONS

- 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

EXAMPLE: ALL ZEROS

Consider:

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

THE USUAL DRILL

```
con  $N : \text{Int} \{0 \leq N\}$   
var  $h : \text{array}[0..N) \text{ of } \text{Int}$   
var  $n : \text{Int}$   
  
 $n := 0$   
 $\{\langle \forall i : 0 \leq i < n : h[i] = 0 \rangle \wedge 0 \leq n \leq N,$   
     $bnd : N - n\}$   
do  $n \neq N \rightarrow ?$   
     $n := n + 1$   
od  
 $\{\langle \forall i : 0 \leq i < N : h[i] = 0 \rangle\}$ 
```

CONSTRUCTING THE LOOP BODY

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

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

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

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

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

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

THE PROGRAM

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

Obvious, but useful.

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

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

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

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

- 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 ≠ N →  
    { establish E = H n }  
    h[n] := E  
    n := n + 1  
od  
{⟨∀i : 0 ≤ i < N : h[i] = H i⟩}
```

- 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.

EXAMPLE: HISTOGRAM

Consider:

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

THE UP LOOP AGAIN

- Let $P\ n$ denote
 $\langle \forall i : 1 \leq i \leq 6 : h[i] = \langle \#k : 0 \leq k < n : X[k] = i \rangle \rangle$.
- A program skeleton:

```
con  $N : \text{Int}$   $\{0 \leq N\}$ ;  $X : \text{array } [0..N) \text{ of } \text{Int}$   
   $\{ \langle \forall i : 0 \leq i < N : 1 \leq X[i] \leq 6 \rangle \}$   
var  $h : \text{array } [1..6] \text{ of } \text{Int}$ ;  $n : \text{Int}$   
  
  initialise  
   $n := 0$   
   $\{ P\ n \wedge 0 \leq n \leq N, \text{bnd} : N - n \}$   
  do  $n \neq N \rightarrow ?$   
     $n := n + 1$   
  od  
   $\{ \langle \forall i : 1 \leq i \leq 6 : h[i] =$   
     $\langle \#k : 0 \leq k < N : X[k] = i \rangle \rangle \}$ 
```

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

$$\begin{aligned} & \langle \forall i : 1 \leq i \leq 6 : h[i] = \langle \#k : 0 \leq k < 0 : X[k] = i \rangle \rangle \\ & \equiv \langle \forall i : 1 \leq i \leq 6 : h[i] = 0 \rangle , \end{aligned}$$

- which can be performed by *allzeros*.

CONSTRUCTING THE LOOP BODY

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

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

- Recall that $\# : \text{Bool} \rightarrow \text{Int}$ is the function such that

$$\begin{aligned} \# \text{ False} &= 0 \\ \# \text{ True} &= 1 \end{aligned}$$

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

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

- Therefore one chooses $l \equiv X[n]$ and $E \equiv h[X[n]] + 1$.

THE PROGRAM

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

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

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

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

var $n : \text{Int}$

$n := 1$

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

$\{P\ 0\}$

$n := 0$

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

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

$n := n + 1$

od

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

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