Programming Languages: Imperative Program Construction 9. Array Manipulation

Shin-Cheng Mu

Autumn Term, 2021

Materials in these notes are mainly from Kaldewaij [Kal90]. Some examples are adapted from the course CSci 550: Program Semantics and Derivation taught by Prof. H. Conrad Cunningham [Cun06], University of Mississippi.

1 Some Notes on Definedness

Assignment Revisited

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

Weakest Precondition

- A more complete rule:

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

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

$$\begin{array}{l} wp\ (\mathbf{if}\ B_0 \rightarrow S_0 \mid B_1 \rightarrow S_1\ \mathbf{fi})\ P = \\ \quad B_0 \Rightarrow wp\ S_0\ P \wedge B_1 \Rightarrow wp\ S_1\ P \wedge (B_0 \vee B_1) \wedge \\ \quad \mathit{def}\ B_0 \wedge \mathit{def}\ B_1\ \ . \end{array}$$

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 $A : \mathbf{array}\ [M..N)\ \mathbf{of}\ \mathit{Int}$ and let I be an expression. We define $\mathit{def}\ (A[I]) = \mathit{def}\ I \wedge M \leqslant I < N$.
- E.g. given $A : \mathbf{array}\ [0..N)\ \mathbf{of}\ \mathit{Int}$,
  - $\mathit{def}\ (A[x\ /\ z] + A[y]) = z \neq 0 \wedge 0 \leqslant x\ /\ z < N \wedge 0 \leqslant y < N$.
  - $wp\ (s := s \uparrow A[n])\ P = P[s \backslash s \uparrow A[n]] \wedge 0 \leqslant n < N$.
- We never made it explicit, because conditions such as $0 \leqslant n < N$ were usually already in the invariant/guard and thus discharged immediately.

2 Array Assignment

- So far, all our arrays have been constants — we read from the arrays but never wrote to them!

- Consider $a : \mathbf{array}\ [0..2)\ \mathbf{of}\ \mathit{Int}$, where $a[0] = 1$ and $a[1] = 1$.
- It should be true that

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

- However, if we use the previous wp,

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

- What went wrong?

Another Counterexample

- For a more obvious example where our previous wp does not work for array assignment:
- $\mathit{wp}\ (a[i] := 0)\ (a[2] \neq 0)$ 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. $a : \mathbf{array}\ [0..N)\ \mathbf{of}\ \mathit{Bool}$ 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?)

Function Alteration

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

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

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

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

wp for Array Assignment

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

$$\begin{aligned}\mathit{wp}\ (a[I] := E)\ P = P[a\backslash(a : I \rightarrow E)] \land&\\ \mathit{def}\ (a[I]) \land \mathit{def}\ E&\ \ .\end{aligned}$$

- In our examples, $\mathit{def}\ (a[I])$ and $\mathit{def}\ E$ 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.

The Example

- Recall our example

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

- We aim to prove

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

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

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

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:

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

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

3 Typical Array Manipulation in a Loop

3.1 All Zeros

Consider:

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

The Usual Drill

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

Constructing the Loop Body

- 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

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

The Program

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

Obvious, but useful.

3.2 Simple Array Assignment

- The calculation can certainly be generalised.
- Given a function $H : \mathit{Int} \rightarrow A$, 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 $P\ n \wedge n \neq N$.
- One can prove the following:

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

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

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

- 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.

3.3 Histogram

Consider:

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

The Up Loop Again

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

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

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

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

- which can be performed by $\mathit{allzeros}$.

Constructing the Loop Body

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

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

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

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

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

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

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

The Program

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

```
$\mathbf{con}\ N : \mathit{Int}\ \{0 \leqslant N\}; X : \mathbf{array}\ [0..N)\ \mathbf{of}\ \mathit{Int}$
$\{\langle \forall i : 0 \leqslant i < N : 1 \leqslant X[i] \leqslant 6 \rangle\}$
$\mathbf{var}\ h : \mathbf{array}\ [1..6]\ \mathbf{of}\ \mathit{Int}$
$\mathbf{var}\ n : \mathit{Int}$
$n := 1$
$\mathbf{do}\ n \neq 7 \rightarrow h[n] := 0; n := n + 1\ \mathbf{od}$
$\{P\ 0\}$
$n := 0$
$\{P\ n \wedge 0 \leqslant n \leqslant N, \mathit{bnd} : N - n\}$
$\mathbf{do}\ n \neq N \rightarrow h[X[n]] := h[X[n]] + 1$
                $n := n + 1$
$\mathbf{od}$
$\{\langle \forall i : 1 \leqslant i \leqslant 6 : h[i] =$
   $\langle \# k : 0 \leqslant k < N : X[k] = i \rangle \rangle\}$
```

References

[Cun06] H. C. Cunningham. CSci 550: Program Semantics and Derivation. `https://john.cs.olemiss.edu/~hcc/csci550/`, 2006.

[Kal90] A. Kaldewaij. *Programming: the Derivation of Algorithms*. Prentice Hall, 1990.