



CSCI 740 - Programming Language Theory

Lecture 24

Verifying Programs with Arrays

Instructor: Hossein Hojjat

October 30, 2017

Weakest Precondition Rules: Summary

c	$\text{wp}(c, Q)$
$x := e$	$Q[x \mapsto e]$
$\text{assume}(b)$	$b \rightarrow Q$
$\text{assert}(b)$	$\text{wp}(b \wedge Q)$
$\text{havoc}(x)$	$\forall y. Q[x \mapsto y]$
$c_1; c_2$	$\text{wp}(c_1, \text{wp}(c_2, Q))$
$\text{if } b \text{ then } c_1 \text{ else } c_2$	$b \rightarrow \text{wp}(c_1, Q) \wedge \neg b \rightarrow \text{wp}(c_2, Q)$
$\text{while } b \text{ do } c$	$I \wedge \forall \vec{y}. \left((I \wedge b \rightarrow \text{wp}(c, I)) \wedge (I \wedge \neg b \rightarrow Q) \right) [\vec{x} \mapsto \vec{y}]$ (\vec{x} are variables modified in c and I is the loop invariant)

Problem with Arrays

```
a[k]=1;  
a[j]=2;  
x=a[k]+a[j];  
{x=3}
```





```
a[k]=1;  
a[j]=2;  
{a[k]+a[j]=3}  
x=a[k]+a[j];  
{x=3}
```



Problem with Arrays

- Now what? Can we use the standard rule for assignment?



$$\text{wp}(x := e, C) = C[x \mapsto e]$$

<code>a[k]=1;</code>		<code>a[k]=1;</code>		<code>a[k]=1;</code>
<code>a[j]=2;</code>		<code>a[j]=2;</code>		<code>a[j]=2;</code>
<code>x=a[k]+a[j];</code>		<code>{a[k]+a[j]=3}</code>		
<code>{x=3}</code>		<code>x=a[k]+a[j];</code>		<code>{x=3}</code>

Problem with Arrays

- Now what? Can we use the standard rule for assignment?

$$\text{wp}(x := e, C) = C[x \mapsto e]$$

<pre>a[k]=1; a[j]=2; x=a[k]+a[j]; {x=3}</pre>		<pre>a[k]=1; a[j]=2; {a[k]+a[j]=3} x=a[k]+a[j]; {x=3}</pre>		<pre>{1+2=3} = {true} a[k]=1; {a[k]+2=3} a[j]=2; {a[k]+a[j]=3} x=a[k]+a[j]; {x=3}</pre>
---	---	---	---	---

Problem with Arrays

- Now what? Can we use the standard rule for assignment?

$$\text{wp}(x := e, C) = C[x \mapsto e]$$

```
a[k]=1;  
a[j]=2;  
x=a[k]+a[j];  
{x=3}
```



```
a[k]=1;  
a[j]=2;  
{a[k]+a[j]=3}  
x=a[k]+a[j];  
{x=3}
```



```
{1+2=3} = {true}  
a[k]=1;  
{a[k]+2=3}  
a[j]=2;  
{a[k]+a[j]=3}  
x=a[k]+a[j];  
{x=3}
```

What if $k = j$?

Problem with Arrays

- Now what? Can we use the standard rule for assignment?

$$wp(x := e, C) = C[x \mapsto e]$$

```
a[k]=1;  
a[j]=2;  
x=a[k]+a[j];  
{x=3}
```



```
a[k]=1;  
a[j]=2;  
{a[k]+a[j]=3}  
x=a[k]+a[j];  
{x=3}
```



```
{1+2=3} = {true}  
a[k]=1;  
{a[k]+2=3}  
a[j]=2;  
{a[k]+a[j]=3}  
x=a[k]+a[j];  
{x=3}
```

What if $k = j$?

Problem with Arrays

- Naïve array assignment axiom does not work

$$\{Q[A[e_1] \mapsto e_2]\} A[e_1] := e_2 \{Q\}$$

- Changes to $A[i]$ may also change $A[j]$, $A[k]$, ...
 - (since i might equal j , k , ...)
- **Solution:** enrich the assertion language with expressions $A\{e_1 \mapsto e_2\}$
- Meaning: the array equal to A except that index e_1 maps to value e_2

$$A\{e_1 \mapsto e_2\}[i] = \begin{cases} A[i] & \text{if } i \neq e_1 \\ e_2 & \text{if } i = e_1 \end{cases}$$

Assignment Rule with Theory of Arrays

$$\frac{}{\vdash \{Q[A \mapsto A\{i \mapsto e\}]\} A[i] := e \{Q\}}$$

```
a[k]=1;  
a[j]=2;  
{a[k]+a[j]=3}  
x=a[k]+a[j];  
{x=3}
```



```
{k≠j}  
{a{k→1}{j→2}[k]+a{k→1}{j→2}[j]=3}  
a[k]=1;  
{a{j→2}[k]+a{j→2}[j]=3}  
a[j]=2;  
{a[k]+a[j]=3}  
x=a[k]+a[j];  
{x=3}
```

Exercise

Prove the array sum is correct

```
{n ≥ 0}
j = 0;
s = 0;
while (j < n) do{
    s = s + a[j];
    j = j + 1;
}
{ s = ∑0 ≤ i < n a[i] }
```

$$\frac{A \Rightarrow I \quad \vdash \{b \wedge I\} c \{I\} \quad I \wedge \neg b \Rightarrow B}{\vdash \{A\} \text{ while } b \text{ do } c \{B\}}$$

Exercise

Prove the array sum is correct

```
{n ≥ 0}
j = 0;
s = 0;
while (j < n) do {
  s = s + a[j];
  j = j + 1;
}
{ s = ∑0 ≤ i < n a[i] }
```

$$\frac{A \Rightarrow I \quad \vdash \{b \wedge I\} c \{I\} \quad I \wedge \neg b \Rightarrow B}{\vdash \{A\} \text{ while } b \text{ do } c \{B\}}$$

Choose invariant $(s = \sum_{0 \leq i < j} a[i]) \wedge 0 \leq j \leq n$

Step 1. Prove invariant is maintained throughout the loop

$$\{j < n \wedge (s = \sum_{0 \leq i < j} a[i]) \wedge 0 \leq j \leq n\}$$

$$s = s + a[j]; \quad j = j + 1$$

$$\{(s = \sum_{0 \leq i < j} a[i]) \wedge 0 \leq j \leq n\}$$

Exercise

Prove the array sum is correct

```
{n ≥ 0}
j = 0;
s = 0;
while (j < n) do {
  s = s + a[j];
  j = j + 1;
}
{ s = ∑0 ≤ i < n a[i] }
```

$$\frac{A \Rightarrow I \quad \vdash \{b \wedge I\} c \{I\} \quad I \wedge \neg b \Rightarrow B}{\vdash \{A\} \text{ while } b \text{ do } c \{B\}}$$

Choose invariant $(s = \sum_{0 \leq i < j} a[i]) \wedge 0 \leq j \leq n$

Step 2. Prove invariant is initially *true*

$$\begin{aligned} & \{n \geq 0\} \\ & j = 0; \quad s = 0 \\ & \{(s = \sum_{0 \leq i < j} a[i]) \wedge 0 \leq j \leq n\} \end{aligned}$$

Exercise

Prove the array sum is correct

```
{n ≥ 0}
j = 0;
s = 0;
while (j < n) do {
  s = s + a[j];
  j = j + 1;
}
{ s = ∑0 ≤ i < n a[i] }
```

$$\frac{A \Rightarrow I \quad \vdash \{b \wedge I\} c \{I\} \quad I \wedge \neg b \Rightarrow B}{\vdash \{A\} \text{ while } b \text{ do } c \{B\}}$$

Choose invariant $(s = \sum_{0 \leq i < j} a[i]) \wedge 0 \leq j \leq n$

Step 3. Prove invariant and exit condition implies postcondition

$$\begin{aligned} ((s = \sum_{0 \leq i < j} a[i]) \wedge 0 \leq j \leq n \wedge j \geq n) &\Rightarrow \\ s &= \sum_{0 \leq i < n} a[i] \end{aligned}$$

Proof Obligations

Step 1. Prove invariant is maintained throughout the loop

$$\{(s + a[j] = \sum_{0 \leq i < j+1} a[i]) \wedge 0 \leq j + 1 \leq n\} \quad (\text{by assignment rule})$$

$$s = s + a[j]$$

$$\{(s = \sum_{0 \leq i < j+1} a[i]) \wedge 0 \leq j + 1 \leq n\} \quad (\text{by assignment rule})$$

$$j = j + 1$$

$$\{(s = \sum_{0 \leq i < j} a[i]) \wedge 0 \leq j \leq n\}$$

Need to show:

$$(0 \leq j \leq n \wedge (s = \sum_{0 \leq i < j} a[i]) \wedge j < n) \Rightarrow \\ (0 \leq j + 1 \leq n \wedge (s + a[j] = \sum_{0 \leq i < j+1} a[i]))$$

Proof Obligations

Step 2. Prove invariant is initially *true*

$$\{(0 = \sum_{0 \leq i < 0} a[i]) \wedge 0 \leq 0 \leq n\} \quad (\text{by assignment rule})$$

$$j = 0$$

$$\{(0 = \sum_{0 \leq i < j} a[i]) \wedge 0 \leq j \leq n\} \quad (\text{by assignment rule})$$

$$s = 0$$

$$\{(s = \sum_{0 \leq i < j} a[i]) \wedge 0 \leq j \leq n\}$$

Need to show:

$$(n \geq 0) \Rightarrow (0 = \sum_{0 \leq i < 0} a[i]) \wedge 0 \leq 0 \leq n$$

Step 3. Prove invariant and exit condition implies postcondition

$$\left((s = \sum_{0 \leq i < j} a[i]) \wedge 0 \leq j \leq n \wedge j \geq n \right) \Rightarrow \\ (s = \sum_{0 \leq i < n} a[i])$$

Exercise

Consider the following program:

```
{0 ≤ i < n}
j = i + 1 ;
while (j < n) {
    a[i] = max(a[i], a[j]);
    j = j + 1;
}
{ ∀ i ≤ k < n a0[k] ≤ a[i] }
```

Is the following a loop invariant?

$$\{\forall i \leq k < j \ a_0[k] \leq a[i] \wedge 0 \leq j \leq n\}$$

(a_0 is the initial array)

Invariant Proof

Prove invariant is maintained throughout the loop

$$\{\forall_{i \leq k < j+1} a_0[k] \leq \max(a[i], a[j]) \wedge 0 \leq j+1 \leq n\}$$

$$\{\forall_{i \leq k < j+1} a_0[k] \leq a\{i \mapsto \max(a[i], a[j])\}[i] \wedge 0 \leq j+1 \leq n\} \text{ (by array assignment)}$$

$$a[i] = \max(a[i], a[j])$$

$$\{\forall_{i \leq k < j+1} a_0[k] \leq a[i] \wedge 0 \leq j+1 \leq n\} \text{ (by assignment)}$$

$$j = j + 1$$

$$\{\forall_{i \leq k < j} a_0[k] \leq a[i] \wedge 0 \leq j \leq n\}$$

Need to show:

$$\begin{aligned} & (\forall_{i \leq k < j} a_0[k] \leq a[i] \wedge j < n) \Rightarrow \\ & (\forall_{i \leq k < j+1} a_0[k] \leq \max(a[i], a[j]) \wedge 0 \leq j+1 \leq n) \end{aligned}$$

We don't know that $a_0[j] \leq \max(a[i], a[j])$!

Conjoin a new constraint $(\forall_{i < k < n} a[k] = a_0[k]) \wedge i < j$

Array Bounds

- Check if an array index is within the bounds of the array

$x := a[i]$

```
assert (0 ≤ i ∧ i < size(a))  
x := a[i]
```

$a[i] := x$

```
assert (0 ≤ i ∧ i < size(a))  
a := a[i ↦ x]
```