Warm-Up Problem

Please fill out your Teaching Evaluation Survey! I will give you a bunch of class time on Thursday.

Real problem: What is the proof rule for Array Assignment? How does it differ from regular assignment?

Recall: The Array-Assignment Rule

Array assignment:

$$\frac{}{(\!|\, Q[A\{e_1 \leftarrow e_2\}/A] \,|\!)\ \texttt{A[e}_1\texttt{] = e}_2\ (\!|\, Q \,|\!)}\text{(Array assignment)}$$

where

$$A\{i \leftarrow e\}[j] = \begin{cases} e, & \text{if } j = i \\ A[j], & \text{if } j \neq i \end{cases} .$$

Program Verification
Arrays: Reversing an Array

Carmen Bruni

Lecture 21

Based on slides by Jonathan Buss, Lila Kari, Anna Lubiw and Steve Wolfman with thanks to B. Bonakdarpour, A. Gao, D. Maftuleac, C. Roberts, R. Trefler, and P. Van Beek

Last Time

- Program Verification: Array rule
- Using the array assignment rule to prove A Hoare triple is satisfied under partial correctness.

Learning Goals

- Complete the example of reversing an array and prove partial correctness.
- Show that reversing an array is totally correct.

Recall

- Last time, you annotated the code for reversing an array.
- Input is $R[1], ..., R[n]$
- For each $1 \leq j \leq \lfloor \frac{n}{2} \rfloor$, we swapped $R[j]$ with $R[n+1-j]$.
- Note throughout, we will implicitly assume that $(n > 0)$. This is so that our array actually has elements.

Code For Reversing an Array

```
$\llparenthesis \left(\forall x \left(((1 \le x) \land (x \le n)) \to (R[x] = r_x)\right)\right) \rrparenthesis$
j = 1 ;
while ( 2*j <= n ) {
    t = R[j] ;
    R[j] = R[n+1-j] ;
    R[n+1-j] = t ;
    j = j + 1 ;
}
$\llparenthesis \left(\forall x \left(((1 \le x) \land (x \le n)) \to (R[x] = r_{n+1-x})\right)\right) \rrparenthesis$
```

The annotation for this code can be found on the next two pages.

Recall

Before, let $\mathrm{Inv}'(j)$ be the formula

$$\Bigg(\forall x \ \Bigg(\Big(\big(((1 \le x) \wedge (x < j)) \big) \rightarrow \big((R[x] = r_{n+1-x}) \wedge (R[n+1-x] = r_x) \big) \Big)$$
$$\wedge \ \Big(\big(((j \le x) \wedge (x \le \tfrac{n+1}{2})) \big) \rightarrow \big((R[x] = r_x) \wedge (R[n+1-x] = r_{n+1-x}) \big) \Big) \Bigg)$$

and further, let

$$R' = R\{j \leftarrow R[((n+1)-j)]\}\{((n+1)-j) \leftarrow R[j]\}$$

Question: Why is this [part of] a reasonable invariant?

Annotation

$\llparenthesis \Big((n > 0) \land \Big(\forall x\,(((1 \leq x) \land (x \leq n)) \to (R[x] = r_x))\Big)\Big) \rrparenthesis$

$\llparenthesis \Big(\mathrm{Inv}'(1) \land (1 \leq (\frac{n}{2} + 1))\Big) \rrparenthesis$ Implied(a)

```
j = 1 ;
```

$\llparenthesis \Big(\mathrm{Inv}'(j) \land (j \leq (\frac{n}{2} + 1))\Big) \rrparenthesis$ Assignment

```
while (2 * j <= n) {
```

  $\llparenthesis \Big(\Big(\mathrm{Inv}'(j) \land (j \leq (\frac{n}{2} + 1))\Big) \land ((2 \cdot j) <= n)\Big) \rrparenthesis$ Partial-Whil

  $\llparenthesis \Big(\mathrm{Inv}'((j+1))[R'/R] \land ((j+1) \leq (\frac{n}{2} + 1))\Big) \rrparenthesis$ Implied(c)

```
    t = R[j]; R[j] = R[n+1-j]; R[n+1-j] = t;
```

  $\llparenthesis \Big(\mathrm{Inv}'((j+1)) \land ((j+1) \leq (\frac{n}{2} + 1))\Big) \rrparenthesis$ Lemma

```
    j = j + 1 ;
```

  $\llparenthesis \Big(\mathrm{Inv}'(j) \land (j \leq (\frac{n}{2} + 1))\Big) \rrparenthesis$ Assignment

```
}
```

$\llparenthesis \Big(\Big(\mathrm{Inv}'(j) \land (j \leq (\frac{n}{2} + 1))\Big) \land ((2 \cdot j) > n)\Big) \rrparenthesis$ Partial-Whil

$\llparenthesis \Big(\forall x\,(((1 \leq x) \land (x \leq n)) \to (R[x] = r_{n+1-x}))\Big) \rrparenthesis$ Implied(b)

Remains To Show

- It remains to prove all of the implied conditions on the previous slide are true (including the lemma).
- We also should complete the prof of total correctness by showing that the while loop terminates.
- We prove the latter first.

Proof that the While Loop Terminates

Consider the loop variant

$$V = \lfloor \frac{n}{2} \rfloor + 1 - j$$

This variant is non-negative and on each iteration of the loop, n remains unchanged while j increments by one. Hence the variant above decrements by 1 each time. Thus, $V \geq 0$ at the beginning since $j = 1$ [note $n > 0$ could be added as well; without this, there would be no elements in the array] and decreases by one during each loop.

Our loop guard is $2j \leq n$ or reworded $j \leq \lfloor \frac{n}{2} \rfloor$. When $V = 0$, we have that $j = \lfloor \frac{n}{2} \rfloor + 1 > \lfloor \frac{n}{2} \rfloor$ and thus the while loop terminates. Hence, our code must terminate.

Proof of Lemma

Why is the lemma true? Hint: Think back to Thursday.

Proof of Lemma

Why is the lemma true? Hint: Think back to Thursday.

The justification for the Lemma is that the three assignment lines simply **swap** the entries $R[j]$ and $R[n+1-j]$ (by the earlier "baby" example of verifying a single swap), and the usual approach to constructing a correct precondition from a given post-condition, with an assignment between.

Proof of Implied (a)

Implied (a) is $(n > 0)$ and the following:

$$\left(\left((\forall x\,((1 \leq x \leq n) \to (R[x] = r_x))) \to \left(\mathrm{Inv}'(1) \wedge \left(1 \leq \frac{n}{2} + 1\right)\right)\right)\right).$$

Proof of Implied (a)

Implied (a) is $(n > 0)$ and the following:

$$\left(\left((\forall x\,((1 \le x \le n) \to (R[x] = r_x))) \to \left(\mathrm{Inv}'(1) \wedge \left(1 \le \frac{n}{2} + 1\right)\right)\right)\right).$$

$\mathrm{Inv}'(1)$ reads

$$\begin{aligned}&(\forall x\,((1 \le x < 1) \to ((R[x] = r_{n+1-x}) \wedge (R[n+1-x] = r_x)))\\ &\wedge \left(\left(1 \le x \le \tfrac{n+1}{2}\right) \to ((R[x] = r_x) \wedge (R[n+1-x] = r_{n+1-x}))\right))).\end{aligned}$$

Proof of Implied (a)

Implied (a) is $(n > 0)$ and the following:

$$\left(\left((\forall x\,((1 \leq x \leq n) \to (R[x] = r_x))) \to \left(\mathrm{Inv}'(1) \wedge \left(1 \leq \frac{n}{2} + 1\right)\right)\right)\right).$$

$\mathrm{Inv}'(1)$ reads

$$\begin{aligned}&(\forall x\,((1 \leq x < 1) \to ((R[x] = r_{n+1-x}) \wedge (R[n+1-x] = r_x)))\\ &\wedge \left(\left(1 \leq x \leq \tfrac{n+1}{2}\right) \to ((R[x] = r_x) \wedge (R[n+1-x] = r_{n+1-x}))))).\end{aligned}$$

Since no x can satisfy $(1 \leq x < 1)$, there is nothing to check in the first implication. The second implication is simply the given precondition re-written, and so the required implication holds.

Proof of Implied (c)

Implied (c) is $\left(\left(\left(\mathrm{Inv}'(j) \land (j \leq (\frac{n}{2}+1))\right) \land ((2 \cdot j) <= n)\right)\right.$

$$\left.\rightarrow \left(\mathrm{Inv}'((j+1))[R'/R] \land ((j+1) \leq (\frac{n}{2}+1))\right)\right)$$

Proof of Implied (c)

Implied (c) is $\left(\left(\left(\mathrm{Inv}'(j) \wedge (j \leq (\frac{n}{2}+1))\right) \wedge ((2 \cdot j) <= n)\right)\right.$
$$\left.\to \left(\mathrm{Inv}'((j+1))[R'/R] \wedge ((j+1) \leq (\frac{n}{2}+1))\right)\right)$$

By construction, R' and R are identical, except at indices j and $n+1-j$. So $\mathrm{Inv}'(j)$ will imply $\mathrm{Inv}'(j+1)[R'/R]$, provided everything is correct in R' at these indices. Everything is clear from the definitions, except possibly the entries $x = j$ and $x = n+1-j$.

- For $(x = j)$ hypothesis is $((R[j] = r_j) \wedge (R[n+1-j] = r_{n+1-j}))$
- For $(x = j)$ conclusion is $((R'[j] = r_{n+1-j}) \wedge (R'[n+1-j] = r_j))$
- By the definition for R', we have

$$\begin{aligned} R'[j] &= R[n+1-j] = r_{n+1-j}, \text{ and} \\ R'[n+1-j] &= R[j] = r_j, \end{aligned}$$

which completes the proof.

Proof of Implied (b)

Recall that Implied (b) is

$$\Big(\Big(\Big(\mathrm{Inv}'(j) \wedge \Big(j \leq (\tfrac{n}{2} + 1)\Big)\Big) \wedge ((2 \cdot j) > n)\Big)$$
$$\to (\forall x\,((1 \leq x \leq n) \to (R[x] = r_{n+1-x})))\Big).$$

Proof of Implied (b)

Recall that Implied (b) is

$$\Bigg(\bigg(\Big(\mathrm{Inv}'(j) \wedge \Big(j \leq (\tfrac{n}{2}+1)\Big)\Big) \wedge ((2 \cdot j) > n)\bigg) \\ \rightarrow (\forall x\,((1 \leq x \leq n) \rightarrow (R[x] = r_{n+1-x})))\Bigg).$$

Recall that $\mathrm{Inv}'(j)$ reads

$$(\forall x\,(1 \leq x < j \rightarrow (R[x] = r_{n+1-x} \wedge R[n+1-x] = r_x)) \\ \wedge (j \leq x \leq \tfrac{n+1}{2} \rightarrow (R[x] = r_x \wedge R[n+1-x] = r_{n+1-x}))).$$

We must analyze $\Big(\Big(j \leq \frac{n}{2}+1\Big) \wedge (2 \cdot j > n)\Big)$ for the cases where n is even and odd.

Case n is even.

If n is even, then $\frac{n}{2}$ is an integer, so that $\left(\left(j \le \frac{n}{2}+1\right) \wedge (2 \cdot j > n)\right)$ gives $j = \frac{n}{2}+1$. Now $\mathrm{Inv}'\left(\frac{n}{2}+1\right)$ reads

$$\begin{aligned}&(\forall x\,(1 \le x < \tfrac{n}{2}+1 \to (R[x] = r_{n+1-x} \wedge R[n+1-x] = r_x))\\ &\wedge (\tfrac{n}{2}+1 \le x \le \tfrac{n+1}{2} \to (R[x] = r_x \wedge R[n+1-x] = r_{n+1-x}))).\end{aligned}$$

Clearly, no x can satisfy the hypothesis of the second implication, so there is nothing to check there. We may rewrite the first implication as

$$\left(1 \le x \le \frac{n}{2} \to \big(R[x] = r_{n+1-x} \wedge R[n+1-x] = r_x\big)\right).$$

This asserts that all the required swaps have been performed, and so the required program post-condition holds, and we are done in this case.

Case n is odd.

If n is odd, then $\frac{n+1}{2}$ is an integer, so that $\left(\left(j \leq \frac{n}{2}+1\right) \wedge (2 \cdot j > n)\right)$, equivalently $\left(\left(j \leq \frac{n+1}{2}+\frac{1}{2}\right) \wedge (2 \cdot j > n)\right)$ gives $j = \frac{n+1}{2}$. Now $\mathrm{Inv}'\left(\frac{n+1}{2}\right)$ reads

$$\begin{aligned}&(\forall x\,(1 \leq x < \tfrac{n+1}{2} \to (R[x] = r_{n+1-x} \wedge R[n+1-x] = r_x))\\ &\wedge (\tfrac{n+1}{2} \leq x \leq \tfrac{n+1}{2} \to (R[x] = r_x \wedge R[n+1-x] = r_{n+1-x}))).\end{aligned}$$

Only $x = \frac{n+1}{2}$ can satisfy the hypothesis of the second implication. Both halves of the $\wedge$-formula in the conclusion of the implication then assert that the middle element was not changed, because $n+1-\frac{n+1}{2} = \frac{n+1}{2}$. We may rewrite the first implication as

$$\left(1 \leq x \leq \frac{n-1}{2} \to \left(R[x] = r_{n+1-x} \wedge R[n+1-x] = r_x\right)\right).$$

This asserts that all the required swaps have been performed, and so the required program post-condition holds, and so we are done in this case.

Example: Binary Search

Binary Search

Binary search is a very common technique, to find whether a given item exists in a sorted array.

Although the algorithm is simple in principle, it is easy to get the details wrong. Hence verification is in order.

Inputs: Array A indexed from 1 to n; integer x.

Precondition: A is sorted: $\forall i\ \forall j\,\Big((1 \le i < j \le n) \to (A[i] \le A[j])\Big)$.

Output values: boolean `found`; integer m.

Postcondition: Either `found` is true and $A[m] = x$, or `found` is false and x does not occur at any location of A.

(Also, A and x are unchanged; We simply won't write to either.)

Code: The outer loop

```
$\llparenthesis\ \forall i\ \forall j\ \big((1 \le i < j \le n) \to (A[i] \le A[j])\big)\ \rrparenthesis$
l = 1;   u = n;   found = false;
$\llparenthesis\ I\ \rrparenthesis$
while ( l <= u and !found ) {
  $\llparenthesis\ I \wedge (l \le u \wedge \neg found)\ \rrparenthesis$                    partial-while
  m = (l+u) div 2;
  $\llparenthesis\ J\ \rrparenthesis$
  if (A[m] = x) {

     ...Body omitted...

  }
  $\llparenthesis\ I\ \rrparenthesis$                                       if-then-else
}
$\llparenthesis\ I \wedge \neg(l \le u \wedge \neg found)\ \rrparenthesis$                      partial-while
$\llparenthesis\ (found \wedge A[m] = x) \vee (\neg found \wedge \forall k\ \neg(A[k] = x))\ \rrparenthesis$
```

Code: the `if`-statement

```
(| $J$ |)
if ( A[m] = x ) {
  (| $J \wedge (A[m] = x)$ |)                       if-then-else
  found = true;
  (| $I$ |)
} else if ( A[m] < x ) {
  (| $J \wedge \neg(A[m] = x) \wedge (A[m] < x)$ |)       if-then-else
  l = m+1;
  (| $I$ |)
} else {
  (| $J \wedge \neg(A[m] = x) \wedge \neg(A[m] < x)$ |)    if-then-else
  u = m - 1;
  (| $I$ |)
}
(| $I$ |)                                           if-then-else
```

Invariant for Binary Search

In the loop, there are two cases:

- We have found the target, at position m.
- We have not yet found the target; if it is present, it must lie beween $A[\ell]$ and $A[u]$ (inclusive).

Expressed as a formula:

$$(\textit{found} \wedge A[m] = x) \vee \Big(\neg\textit{found} \wedge \forall i \Big((A[i] = x) \to (\ell \le i \le u)\Big)\Big) \ .$$

It turns out that the above is more specific than necessary. As the actual invariant, we shall use the formula

$$I = (\textit{found} \to A[m] = x) \wedge \Big(\forall i \Big((A[i] = x) \to (\ell \le i \le u)\Big)\Big) \ .$$

(*Exercise: As you go through the proof, check what would happen if we used the first formula instead.*)

Annotations for `while`

$(\!|\ \forall i\ \forall j \left((1 \le i < j \le n) \rightarrow (A[i] \le A[j])\right)\ |\!)$

```
l = 1;   u = n;   found = false;
```

$(\!|\ (\mathit{found} \rightarrow A[m] = x) \wedge \left(\forall i \left((A[i] = x) \rightarrow (\ell \le i \le u)\right)\right)\ |\!)$

```
while ( l <= u && !found ) {
```

$(\!|\ (\mathit{found} \rightarrow A[m] = x) \wedge \left(\forall i \left((A[i] = x) \rightarrow (\ell \le i \le u)\right)\right)$ partial-

$\wedge\ (l \le u) \wedge \neg \mathit{found}\ |\!)$ while

$(\!|\ \forall i \left((A[i] = x) \rightarrow (\ell \le i \le u)\right) \wedge \neg \mathit{found} \wedge (\ell \le \lfloor(\ell + u)/2\rfloor \le u)\ |\!)$

implied

```
  m = (l+u) div 2 ;
```

$(\!|\ \forall i \left((A[i] = x) \rightarrow (\ell \le i \le u)\right) \wedge \neg \mathit{found} \wedge (\ell \le m \le u)\ |\!)$ assignment

The last condition is the formula "J": the precondition for the `if`-statement.

First Branch of the `if`-Statement

$(\!|\ \forall i \left((A[i] = x) \to (\ell \le i \le u)\right) \land \neg \textit{found} \land (\ell \le m \le u)\ |\!)$

```
if ( A[m] = x ) {
```

$\quad(\!|\ \forall i \left((A[i] = x) \to (\ell \le i \le u)\right) \land \neg \textit{found} \land (\ell \le m \le u) \land (A[m] = x)\ |\!)$

if-then-else

$\quad(\!|\ (\textit{true} \to A[m] = x) \land \left(\forall i \left((A[i] = x) \to (\ell \le i \le u)\right)\right)\ |\!)$ implied

```
  found = true;
```

$\quad(\!|\ (\textit{found} \to A[m] = x) \land \left(\forall i \left((A[i] = x) \to (\ell \le i \le u)\right)\right)\ |\!)$ assignment

```
}
```

The implication is trivial.

Second Branch of the `if`-Statement

$\llparenthesis\ \forall i \left((A[i]=x) \rightarrow (\ell \le i \le u)\right) \wedge \neg \mathit{found} \wedge (\ell \le m \le u) \wedge \neg(A[m]=x)\ \rrparenthesis$

```
if ( A[m] < x ) {
```

$\llparenthesis\ \forall i \left((A[i]=x) \rightarrow (\ell \le i \le u)\right) \wedge \neg \mathit{found} \wedge (\ell \le m \le u)$
$\wedge \neg(A[m]=x) \wedge (A[m] < x)\ \rrparenthesis$ if-then-else

$\llparenthesis\ (\mathit{found} \rightarrow A[m]=x) \wedge \left(\forall i \left((A[i]=x) \rightarrow (m+1 \le i \le u)\right)\right)\ \rrparenthesis$ implied

```
  l = m + 1;
```

$\llparenthesis\ (\mathit{found} \rightarrow A[m]=x) \wedge \left(\forall i \left((A[i]=x) \rightarrow (\ell \le i \le u)\right)\right)\ \rrparenthesis$ assignment

```
}
```

To justify the implication, show that $A[j] < x$ whenever $\ell \le j \le m$.

This follows from the condition that A is sorted, together with $A[m] < x$.