Program Verification
Array Assignments

Alice Gao
Lecture 21

Based on work by J. Buss, L. Kari, A. Lubiw, B. Bonakdarpour, D. Maftuleac, C. Roberts, R. Trefler, and P. Van Beek
Outline

Program Verification: Array Assignments
Learning Goals
Introducing the array assignment rule
An example using the array assignment rule
Revisiting the Learning Goals
Learning Goals

By the end of this lecture, you should be able to:
Partial correctness for array assignments

▸ Prove that a Hoare triple is satisfied under partial correctness for a program containing array assignment statements.
Let \( A \) be an array of \( n \) integers.

Consider the following triple. What should the precondition be?

\[
\begin{align*}
???
A[x] &= 1; \\
A[y] &= 0
\end{align*}
\]

array assignment

- If \( x = y \), the precondition should be ...?
- If \( x \neq y \), the precondition should be ...?

We are using variables as indices into arrays. We must consider multiple cases for all possible values of the variables.
The array assignment inference rule

Let $A$ be an array of $n$ integers.

First, write down the sequence of changes. Resolve all of the changes when we prove the implied’s.

$$Q[A\{e_1 \leftarrow e_2\}/A]$$

$A[e_1] = e_2$;

$Q$ array assignment

- $A$ is the original array.
- $A\{e_1 \leftarrow e_2\}$ is the new array, which is identical to array $A$ except that the $e_1^{th}$ element is $e_2$. 
The array re-assignment notation:

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

Note that \(e_1\) is an index whereas \(e_2\) is an array element.

We apply assignments from left to right.

**Examples:**

- \(A\{1 \leftarrow 3\}[1] = 3\)
- \(A\{1 \leftarrow 3\}\{1 \leftarrow 4\}[1] = 4\)
CQ 1 Applying the array assignment rule

CQ 1: What is the precondition derived using the array assignment inference rule?

???
A[1] = 2;
A[x] = y0  array assignment

(A) A{1 ← 1}[x] = y0
(B) A{1 ← 2}[x] = y0
(C) A{2 ← 1}[x] = y0
(D) A{2 ← 2}[x] = y0
(E) None of the above
CQ 2: What is the precondition derived using the array assignment inference rule?

\[ A[1] = 2; \]
\[ A\{3 \leftarrow 4\}[x] = y_0 \]

(A) \[ A\{1 \leftarrow 2\}\{3 \leftarrow 4\}[x] = y_0 \]
(B) \[ A\{3 \leftarrow 4\}\{1 \leftarrow 2\}[x] = y_0 \]
(C) None of the above
CQ 3 Applying the array assignment rule

CQ 3: What is the precondition derived using the array assignment inference rule?

???

\[ A[1] = 2; \]
\[ A\{3 \leftarrow A[y]\}[x] = y0 \quad \text{array assignment} \]

(A) \[ A\{1 \leftarrow 2\}\{3 \leftarrow A[y]\}[x] = y0 \]
(B) \[ A\{1 \leftarrow 2\}\{3 \leftarrow A\{1 \leftarrow 2\}[y]\}[x] = y0 \]
(C) None of the above
Example of the array assignment rule

Example:
Prove that the following triple is satisfied under partial correctness.

\[ ((A[x] = x0) \land (A[y] = y0)) \]

\[ t = A[x]; \]

\[ A[x] = A[y]; \]

\[ A[y] = t; \]

\[ ((A[x] = y0) \land (A[y] = x0)) \]
Reversing an array

Consider an array $R$ of $n$ integers, $R[1], R[2], ..., R[n]$.

We want to reverse the order of its elements.

Our algorithm:

For each $1 \leq j \leq \lfloor n/2 \rfloor$, we will swap $R[j]$ with $R[n + 1 - j]$. 

Reversing an array

\( R \) is an array of \( n \) integers, \( R[1], R[2], \ldots, R[n] \). Prove that the following triple is satisfied under partial correctness.

\[
(\forall x ((1 \leq x \leq n) \rightarrow (R[x] = r_x)))
\]

\( j = 1; \)

\textbf{while} (2 \ast j <= n) {
  \[
  t = R[j];
  \]
  \[
  R[j] = R[n+1-j];
  \]
  \[
  R[n+1-j] = t;
  \]
  \[
  j = j + 1;
  \]
}

\[
(\forall x ((1 \leq x \leq n) \rightarrow (R[x] = r_{n+1-x})))
\]
Reversing an array

$R$ is an array of $n$ integers, $R[1], R[2], ..., R[n]$. Prove that the following triple is satisfied under partial correctness.

Let $Inv(j)$ denote our invariant.

\[
(\forall x ((1 \leq x \leq n) \rightarrow (R[x] = r_x)))
\]

\[
j = 1;
\]

```
while (2 * j <= n) {
    t = R[j];
    R[j] = R[n+1-j];
    R[n+1-j] = t;
    j = j + 1;
}
```

\[
(\forall x ((1 \leq x \leq n) \rightarrow (R[x] = r_{n+1-x})))
\]
Revisiting the learning goals

By the end of this lecture, you should be able to:
Partial correctness for array assignments

- Prove that a Hoare triple is satisfied under partial correctness for a program containing array assignment statements.