Array Element Assignments

A. Why?

- Array assignments aren't like assignments to plain variables because the actual item to change can't be determined until runtime. We can handle this by extending our notion of assignment and/or substitution.

B. Outcomes

After this class, you should

- Know how to perform textual substitution to replace an array element.
- Know how to calculate the \( wp \) of an array element assignment.

C. Array Element Assignments

- An array assignment \( b[e_0] := e_1 \) (where \( e_0 \) and \( e_1 \) are expressions) is different from a plain variable assignment because the exact element being changed may not be known at program annotation time. E.g., compare these two triples:
  
  - **Valid**: \( \{T\} x := y; y := y + 1 \ \{x < y\} \)
  
  - **Invalid**: \( \{T\} b[k] := b[j]; b[j] := b[j] + 1 \ \{b[k] < b[j]\} \)

- The problem is what happens if \( k = j \) at runtime: What is \( wp(b[j] := b[j] + 1, b[k] < b[j]) \)?

- The answer should be something like

  "If \( k \neq j \) then \( b[k] < b[j] + 1, \) else \( b[j] + 1 < b[j] + 1 \) (which is false)"

- There are two alternatives for handling array assignments

  - The alternative we'll use involves defining the \( wp \) of an array assignment using an extended notion of textual substitution:

    \[
    wp(b[e_0] := e_1, p) = p[e_1 / b[e_0]] \text{ and } \{p[e_1 / b[e_0]] \ b[e_0] := e_1 \ \{p\}\}
    \]

  - Of course, we need to figure out what syntactic substitution for an array indexing expression means: \( (predicate)[expression/b[e_0]] \)

  - Side note: The other way to handle array assignments, the Dijkstra / Gries technique, is to introduce a new kind of expression and view the array assignment \( b[e_0] := e_1 \) as short for \( b := \) this new kind of expression.

- If we have time later (we probably won't), we'll study this.
D. Substitution for Array Elements

- We'll need to substitute into expressions and predicates. We'll tackle expressions first; below, $b_1$ and $b_2$ are different arrays.
  - $(b_2[e_2])[e/b_1[e_1]]$
  - $(b_1[e_2])[e/b_1[e_1]]$

- If $b_1$ and $b_2$ are different array names then substituting into $b_2[e_2]$ will only require us to look at substituting into $e_2$:
  
  $$(b_2[e_2])[e/b_1[e_1]] = b_2[e_2'] \text{ where } e_2' = (e_2)[e/b_1[e_1]]$$

- When the the array names match, as in $(b[k])[e'/b[e]]$, we have to check the indexes $k$ and $e$ for equality at runtime; to do that, we can use a conditional expression.

  • **Definition, case 1:** $(b[k])[e_1/b[e]] = (\text{if } k = e \text{ then } e_1 \text{ else } b[k] \text{ fi})$.
    - If $k = e$ at runtime, then $(b[k])[e_1/b[e]] = e_1$.
    - If $k \neq e$ at runtime, then $(b[k])[e_1/b[e]] = b[k]$.

  • **Example 1:** $(b[k])[5/b[0]] = (\text{if } k = 0 \text{ then } 5 \text{ else } b[k] \text{ fi})$.

  • **Example 2:** $(b[k])[e_1/b[j]] = (\text{if } k = j \text{ then } e_1 \text{ else } b[k] \text{ fi})$.

  • **Example 3:** $(b[k])[b[j]+1/b[j]] = (\text{if } k = j \text{ then } b[j]+1 \text{ else } b[k] \text{ fi})$.
    - Note: In $(b[k])[e_1/b[e_2]]$, we don't substitute into $e_1$, even if it involves $b$.

  • **Example 4:** $(b[k])[b[i]/b[j]] = (\text{if } k = j \text{ then } b[i] \text{ else } b[k] \text{ fi})$.

The General Case for Array Element Substitution

- When $e_2$ is not just a simple variable or constant, then in $(b[e_2])[e/b[e_1]]$, we have to check $e_2$ for uses of $b[...]$ and substitute for them also.

  • **Definition, case 2:**
    
    $$(b[e_2])[e_0/b[e_1]] = (\text{if } e_2' = e_1 \text{ then } e_0 \text{ else } b[e_2'] \text{ fi}) \text{ where } e_2' = (e_2)[e_0/b[e_1]].$$

  - This subsumes the earlier case, since if $e_2 = k$ then $e_2' = k[e_0/b[e_1]] = k$. We get
    
    $$(b[k])[e/b[e_1]] = (\text{if } k = e_1 \text{ then } e \text{ else } b[k] \text{ fi})$$

Example 5

- Consider $(b[b[k]][5/b[0]])$ — how should it behave? The nested $b[k]$ should behave like $5$ if $k = 0$, otherwise it's behaves like $b[k]$ as usual. The outer $b[...]$ should behave like $5$ if its index behaves like $0$, otherwise it should behave as $b[its\ index]$.

- Following the definition above, we get
  
  $$(b[b[k]][5/b[0]]) = (\text{if } e_2' = 0 \text{ then } 5 \text{ else } b[e_2'] \text{ fi})$$
  
  where $e_2' = (b[k])[5/b[0]] = (\text{if } k = 0 \text{ then } 5 \text{ else } b[k] \text{ fi})$
Substituting the (textual) value of $e_2'$ gives us

\[(b[b[k]][5 / b[0]]) = \text{if (if } k = 0 \text{ then } 5 \text{ else } b[k] \text{ fi) = 0} \]

\[\text{then } 5 \]

\[\text{else } b[\text{if } k = 0 \text{ then } 5 \text{ else } b[k] \text{ fi) fi} \]

In the next section, we'll see how to simplify expressions so that we like this to get

\[\text{if } k = 0 \text{ then } b[5] \text{ else if } b[k] = 0 \text{ then } 5 \text{ else } b[b[k]] \text{ fi fi} \]

E. Optimization of Static Cases

Because $e[e_1 / b[e_0]]$ can result in a complicated piece of text, it can be useful to shorten it using various optimizations, similarly to how compilers can optimize code.

All the optimizations below are intended to be done “statically” (at compile time) — we inspect the text of an expression before the code ever runs.

For the easiest examples, if we know whether $k = e_0$ or not, then we can optimize $\text{if } k = e_0 \text{ then } ... \text{ else fi to just the true branch or the false branch.}$

General principle: If we know statically that $k = e'$, then we can optimize $(b[k])[e'/b[e]] = \text{if } k = e \text{ then } e' \text{ else } b[k] \text{ fi to } e'$. If we know $k \neq e'$, then we can optimize to $b[k]$.

Notation: $e_1 \mapsto e_2$ ("$e_1$ optimizes to $e_2"$) means expression $e_1$ can be replaced by expression $e_2$.

Example 6: $(b[0])[e_1 / b[2]] = \text{if } 2 = 2 \text{ then } e_1 \text{ else } b[0] \text{ fi } \mapsto b[0]$.

Example 7: $(b[2])[e_1 / b[2]] = \text{if } 2 = 2 \text{ then } e_1 \text{ else } b[2] \text{ fi } \mapsto e_1$.

Example 8:

* $(b[0])[e'/b[1]] = \text{if } 0 = 1 \text{ then } e' \text{ else } b[0] \text{ fi } \mapsto b[0]$.
* $(b[1])[e'/b[1]] = \text{if } 1 = 1 \text{ then } e' \text{ else } b[1] \text{ fi } \mapsto e'$.
* $(b[1])[3 / b[2]] = \text{if } 1 = 2 \text{ then } 3 \text{ else } b[1] \text{ fi } \mapsto b[1]$.
* $(b[x])[e'/b[x]] = \text{if } x = x \text{ then } e' \text{ else } b[x] \text{ fi } \mapsto e'$.

F. Rules for Simplifying Conditional Expressions

Let's identify some general rules for simplifying conditional expressions and predicates involving them. This will let us simplify calculation of $wp$ for array assignments.

* $(\text{if } T \text{ then } e_1 \text{ else } e_2 \text{ fi}) \mapsto e_1$
* $(\text{if } F \text{ then } e_1 \text{ else } e_2 \text{ fi}) \mapsto e_2$
* $(\text{if } B \text{ then } e \text{ else } e \text{ fi}) \mapsto e$

* If $(B \Rightarrow e_1 = e_2)$, then $(\text{if } B \text{ then } e_1 \text{ else } e_2 \text{ fi}) \mapsto e_2$
* If $(\neg B \Rightarrow e_1 = e_2)$, then $(\text{if } B \text{ then } e_1 \text{ else } e_2 \text{ fi}) \mapsto e_1$

Let $\circ$ be a unary operator or relation and $\oplus$ be a binary operation or relation

* $\circ(\text{if } B \text{ then } e_1 \text{ else } e_2 \text{ fi}) \mapsto (\text{if } B \text{ then } \circ e_1 \text{ else } \circ e_2 \text{ fi})$
• (if \(B\) then \(e_1\ else \(e_2\ \text{fi}) \oplus e_3 \rightarrow (if \(B\) then \(e_1\ \oplus e_3\ else \(e_2\ \oplus e_3\ \text{fi})

• \(b[\text{if } B \text{ then } e_1\ else e_2\ \text{fi}] \rightarrow \text{if } B \text{ then } b[e_1] else b[e_2]\ \text{fi}

• For any function \(f(...), f(\text{if } B \text{ then } e_1\ else e_2\ \text{fi}) \rightarrow \text{if } B \text{ then } f(e_1) else f(e_2)\ \text{fi}

• If \(B, B_1\), and \(B_2\) are boolean expressions, then

  • (if \(B\) then \(B_1\ else F\ \text{fi}) \Leftrightarrow (B \land B_1)
  • (if \(B\) then \(F\ else B_2\ \text{fi}) \Leftrightarrow (\neg B \land B_2)
  • (if \(B\) then \(B_1\ else T\ \text{fi}) \Leftrightarrow (B \rightarrow B_1) \Leftrightarrow (\neg B \lor B_1)
  • (if \(B\) then \(T\ else B_2\ \text{fi}) \Leftrightarrow (\neg B \rightarrow B_2) \Leftrightarrow (B \lor B_2)
  • (if \(B\) then \(B_1\ else B_2\ \text{fi}) \Leftrightarrow ((B \rightarrow B_1) \land (\neg B \rightarrow B_2)) \Leftrightarrow ((B \land B_1) \lor (\neg B \land B_2)).

**Example 9:**

\[wp(b[j] := b[j]+1, b[k] < b[j]) = (b[k] < b[j] \land b[j] < 1/b[j]) = (b[k] \land b[j]+1/b[j]) < (b[j]) \land b[j]+1/b[j] = if \ k = j \ then \ b[j]+1 \ else \ b[k] \ fi < b[j]+1 \]

\[\Leftrightarrow \ if \ k = j \ then \ b[j]+1 < b[j]+1 \ else \ b[k] < b[j]+1 \ fi \]

\[\Leftrightarrow \ if \ k = j \ then \ F \ else \ b[k] < b[j]+1 \ fi \]

\[\Leftrightarrow \ k \neq j \ and \ b[k] < b[j]+1 \]

This gives us the following correctness triple:

\[\{k \neq j \land b[k] < b[j]+1\} \rightarrow b[j] := b[j]+1 \{b[k] < b[j]\}\]

**G. Swapping Array Elements**

• To illustrate the use of array references, let’s look at the problem of swapping array elements.

• To swap simple variables \(x\) and \(y\) using a temporary variable \(u\), we can use logical variables \(c\) and \(d\) and prove

  \[
  \{x = \text{c} \land y = \text{d}\} \ u := x; \ x := y; \ y := u \ \{x = \text{d} \land y = \text{c}\}
  \]

• We can prove this program correct by expanding to a full proof outline; here we’re using \(wp\).

  \[
  \{x = \text{c} \land y = \text{d}\}
  \{y = \text{d} \land x = \text{c}\} \ u := x;
  \{y = \text{d} \land u = \text{c}\} \ x := y;
  \{x = \text{d} \land u = \text{c}\} \ y := u
  \{x = \text{d} \land y = \text{c}\}
  \]

• **Example 10:** For swapping \(b[m]\) and \(b[n]\), we want to prove

  \[
  \{b[m] = \text{c} \land b[n] = \text{d}\} \ u := b[m]; \ b[m] := b[n]; \ b[n] := u \ \{b[m] = \text{d} \land b[n] = \text{c}\}
  \]

As with simple variables, we can prove this holds by using \(wp\) to expand to the full proof outline. Let \(p = b[m] = \text{c} \land b[n] = \text{d}\) and \(q = b[m] = \text{d} \land b[n] = \text{c}\), then we can prove

\[
\{p\} \ \{q_3\} \ u := b[m]; \ \{q_2\} \ b[m] := b[n]; \ \{q_1\} \ b[n] := u \ \{q\}
\]
by using

\* $q_1 = wp(b[n] := u, q) = q[u/b[n]]$
\* $q_2 = wp(b[m] := b[n], q_1) = q_1[b[n]/b[m]]$
\* $q_3 = wp(u := b[m], q_2) = q_2[b[m]/u]$
\* (and hopefully) $p \rightarrow q_3$

We'll do this in steps.

\* $q_1 = q[u/b[n]]$
  \* $= (b[m] = d \land b[n] = c) \backslash u[b[n]]$
  \* $= (b[m] = d) \land (b[n] = c)[u[b[n]]$
  \* $= (b[m])[u/b[n]] = d \land (b[n])[u/b[n]] = c$
  \* $= (if \ m = n \ then \ u \ else \ b[m]) \fi = d \land u = c$  
    // Stop here for a purely syntactic result

\* $q_2 = q_1[b[n]/b[m]]$
  \* $= ((if \ m = n \ then \ u \ else \ b[m]) \fi = d \land u = c) \backslash b[n]/b[m]]$
  \* $= (if \ m = n \ then \ u \ else \ b[m]) \backslash b[n]/b[m]] \fi = d \land u = c$
  \* $= (if \ m = n \ then \ u \ else \ b[n]) \fi = d \land u = c$

\* $q_3 = q_2[b[m]/u]$
  \* $= ((if \ m = n \ then \ u \ else \ b[n]) \fi = d \land u = c)[b[m]/u]$
  \* $= (if \ m = n \ then \ b[m] \ else \ b[n]) \fi = d \land b[m] = c$
    // Continuing with logical manipulation
    $\iff (if \ m = n \ then \ b[n] \ else \ b[n]) \fi = d \land b[m] = c$  
    // if $m = n$ then $b[m] = b[n]$
    $\iff b[n] = d \land b[m] = c$

\* Since $p = b[m] = c \land b[n] = d$, we get $p \rightarrow q_3$. (End of Example 10)
Array Element Assignments

A. Why?

• Array assignments aren't like assignments to plain variables because the actual item to change can't be determined until runtime. We can handle this by extending our notion of assignment and/or substitution.

B. Outcomes

At the end of this work you should:

• Be able to perform textual substitution to replace an array element.
• Be able to calculate the wp of an array element assignment.

C. Questions

For each of the questions below, calculate the given weakest precondition. Then try logically simplifying it to something easier to read.

1. Calculate \( wp(b[0] := 9, x > b[k]) \).

2. Calculate \( wp(b[k] := b[j], b[m] = 0) \).

3. Calculate \( wp(b[k] := b[j], b[j] = z) \).

4. Calculate \( wp(b[k] := 1, b[k] = b[j]) \).

5. Calculate \( wp(b[k] := x; b[j] := y, b[k] \neq b[j]) \).

6. Calculate \( wp(b[k] := x, b[b[k]] \neq b[k]) \).

7. Is the triple \( \{k < b[k] < b[j]\} b[b[k]] := b[j] \{b[k] \neq b[j]\} \) valid?

8. Define a predicate function \( swapped(b_1, b_2, k, j) \) that yields true iff \( b_1 \) and \( b_2 \) are equal except their values at \( k \) and \( j \) are swapped.
9. Let the function \( \text{search}(b, m, n, x) = k \) where if \( m \leq k \leq n \) then \( b[k] = x \) and if \( k < m \lor k > n \), then no index for \( x \) exists in the range \( m \ldots n \). (I.e., for all \( j \) where \( m \leq j \leq n \), we have \( b[j] \neq x \).) Complete the predicate \( \exists k \cdot \text{search}(b, m, n, x) = k \rightarrow \ldots \) so that the ellipsed section formalizes this description of what \( \text{search} \) does.
Solution to Practice 21 (Array Element Assignments)

1. $wp(b[0] := 9, x > b[k]) = (x > b[k])[9/b[0]]$
   
   $= x > \text{if } k = 0 \text{ then } 9 \text{ else } b[k] \text{ fi}$
   
   $\iff k = 0 \text{ then } x > 9 \text{ else } x > b[k] \text{ fi}$
   
   $\iff (k = 0 \land x > 9) \lor (k \neq 0 \land x > b[k])$

2. $wp(b[k] := b[j], b[m] = 0) = (b[m] = 0)[b[j]/b[k]]$
   
   $= \text{if } m = k \text{ then } b[j] \text{ else } b[m] \text{ fi}$
   
   $\iff m = k \lor (m \neq k \land b[m] = 0)$
   
   $\iff m = k \land b[j] = 0 \lor m \neq k \land b[m] = 0$ (another possible alternative rewriting)

3. $wp(b[k] := b[j], b[j] = z) = (b[j] = z)[b[j]/b[k]]$
   
   $= \text{if } j = k \text{ then } b[j] \text{ else } b[j] \text{ fi}$
   
   $\iff j = k \lor (j \neq k \land b[j] = z)$ (one possible alternative rewriting)
   
   $\iff j = k \land b[j] = z \lor j \neq k \land b[j] = z$ (another possible alternative rewriting)

4. $wp(b[k] := 1, b[k] = b[j]) = (b[k] = b[j])[1/b[k]]$
   
   $= (b[k])[1/b[k]] = (b[j])[1/b[k]]$
   
   $= 1 = (\text{if } j = k \text{ then } 1 \text{ else } b[j] \text{ fi})$
   
   $\iff k \lor b[j] = 1$ (one possible alternative rewriting)
   
   $\iff k \neq j \lor b[j] = 1$ (if you prefer $\land$ to $\lor$)

5. $wp(b[k] := x; b[j] := y, b[k] \neq b[j])$
   
   $= wp(b[k] := x, wp(b[j] := y, b[k] \neq b[j])]$
   
   For the embedded $wp$, $wp(b[j] := y, b[k] \neq b[j])$
   
   $= (b[k] \neq b[j])[y/b[j]]$
   
   $= b[k][y/b[j]] \neq b[j][y/b[j]]$
   
   $= \text{if } k = j \text{ then } y \text{ else } b[k] \text{ fi} \neq y$
   
   $\iff k \neq j \land b[k] \neq y$

   So $wp(b[k] := x, wp(b[j] := y, b[k] \neq b[j])$

   $\iff wp(b[k] := x, k \neq j \land b[k] \neq y)$
   
   $= (k \neq j \land b[k] \neq y)[x/b[k]]$
   
   $= k \neq j \land b[k][x/b[k]] \neq y$
   
   $= k \neq j \land x \neq y$

   Intuitively, if $k = j$, then we can't have $b[k] \neq b[j]$. Even if $k \neq j$, we need $x \neq y$ to ensure that the assignments $b[k] := x; b[j] := y$ make $b[k] \neq b[j]$.

6. For $wp(b[k] := x, b[b[k]] \neq b[k])$, let's first look at substituting $x$ for $b[k]$ in $b[b[k]]$. It's complicated because we have to recursively substitute in the index of the outer $b[...]$. The general rule is

   $(b[e_2])[e_1/b[e_0]] = (\text{if } e_2 = e_0 \text{ then } e_1 \text{ else } b[e_2'] \text{ fi})$ where $e_2' = (e_2)[e_1/b[e_0]]$

   So let $e' = (b[k])[x/b[k]] = x$, then $(b[b[k]])[x/b[k]]$

   $= \text{if } e' = k \text{ then } x \text{ else } b[e'] \text{ fi}$

   $= \text{if } x = k \text{ then } x \text{ else } b[x] \text{ fi}$
Then \( wp(b[k] := x, b[k] \neq b[k]) \)
\( = (b[k] \neq b[k]) \cdot x / b[k] \)
\( = (b[k] \neq b[k]) \cdot x / b[k] \)
\( = \text{if } x = k \text{ then } x \text{ else } b[x] fi \neq x \)
\( \Leftrightarrow x \neq k \land b[x] \neq x \)

7. For the triple to be valid, it's sufficient to show that its precondition implies the \( wp \) of the assignment and postcondition. I.e., \( k < b[k] < b[j] \rightarrow wp(b[b[k]] := b[j], b[k] \neq b[j]) \)

First let's calculate \( wp(b[b[k]] := b[j], b[k] \neq b[j]) \)
\( = (b[k] \neq b[j]) \cdot b[j] / b[k] \)
\( = (b[k] \neq b[j]) \cdot b[j] / b[k] \)
\( = \text{if } k = b[k] \text{ then } b[j] \text{ else } b[k] fi \neq \text{if } j = b[k] \text{ then } b[j] \text{ else } b[j] fi \)
\( \Leftrightarrow \text{if } k = b[k] \text{ then } b[j] \text{ else } b[k] fi \neq \text{j} \)
\( \Leftrightarrow \text{if } k = b[k] \text{ then } b[j] \neq b[j] \text{ else } b[k] \neq b[j] fi \)
\( \Leftrightarrow k \neq b[k] \land b[k] \neq b[j] \)

Going back to our original question, if the implication below is valid, then our triple is valid (because we have the precondition of the triple implying the \( wp \) of its body and postcondition).
\( k < b[k] < b[j] \rightarrow wp(b[b[k]] := b[j], b[k] \neq b[j]) \)
\( \Leftrightarrow k < b[k] < b[j] \rightarrow k \neq b[k] \land b[k] \neq b[j] \)
\( \Leftrightarrow \top \)

So our original triple is indeed valid.

8. \( \text{swapped}(b_1, b_2, k, j) = b_1[k] = b_2[j] \land b_1[j] = b_2[k] \land (\forall m. (m \neq k \land m \neq j \rightarrow b_1[m] = b_2[m])) \)

9. \( \exists k. \text{search}(b, m, n, x) = k \rightarrow (m \leq k \leq n \rightarrow b[k] = x) \land (m > k \lor k > n \rightarrow (\forall j. m \leq j \leq n \rightarrow b[j] \neq x)) \)
(Extra parentheses added for readability)