Convergence; Finding Invariants; Array Assignments

CS 536: Science of Programming, Spring 2018

Due Mon Apr 16, 11:59 pm

4/19 solved

A. *Instructions*

- You can work together in groups of ≤ 4. Submit your work on Blackboard. Submit one copy, under the name of one person in the group (doesn't matter who). Include the names and A-IDs of everyone in the group (including the submitter) inside that copy.

B. *Why?*

- To avoid runtime errors, we use domain predicates; to avoid infinite loops, we use bound functions.
- There is no algorithm for finding loop invariants, but there are some heuristics.
- We can handle array assignments if we extend our notion of substitution.

C. *Outcomes*

After this homework, you should be able to

- State the conditions for proving loop convergence; use heuristics for finding bound functions for simple loops.
- Generate possible invariants using heuristics.
- Perform textual substitution to replace an array element and calculate the *wp* of an array element assignment.

D. *Problems [100 points total]*

1. [12 = 6 * 2 points] Expand the minimal outline below into a full proof outline for total correctness. You'll need to include the implicit **else skip**, add domain predicate(s) $D(\ldots)$, and define p_0.

   $\{p_0\}$ **if** b[j] ≥ 0 **then** x := sqrt(b[j]) **fi** {x = sqrt(b[j])}

2. [15 = 5 * 3 points] Consider the loop {**inv** *p*} {**bd** *t*} **while** j ≤ n **do** … j := j+1 **od**. If $(p \rightarrow n \geq 0 \wedge 0 < C \leq j \leq n+C)$ (where C is a named constant), then for each of the following expressions, say whether or not it can be used as the bound expression *t* above (if not, briefly explain why).

   a. n

   b. n – j

   c. n – j + C

   d. n + j + C

   e. n – j + 2*C

3. [24 = 8 * 3 points] Expand the minimal outline below into a full proof outline for total correctness. (I.e., avoid runtime errors and loop divergence.) As part of this, define the initial precondition p_0 and initialization code S_0. The loop invariant is partially defined (extend it if you need to). Feel free to define other predicates if you find it helpful. Include a list of predicate logic obligations and show the expansion of any substitutions.

   $\{p_0\}\ S_0$ **;**
   $\{$**inv** $p \equiv \texttt{x} = \texttt{2^k} \le \texttt{b[j]} \land\ ???\ \}$ $\{$**bd** $???\}$
   while $\texttt{2*x} \le \texttt{b[j]}$ **do**
   $\quad$ `k:= k+1;`
   $\quad$ `x := 2*x`
   od
   $\{\texttt{x} = \texttt{2^k} \le \texttt{b[j]} < \texttt{2^(k+1)}\}$

For Problems 4 and 5, Let $q \equiv \texttt{x} \ge 0 \land \texttt{z} = \texttt{2^x} \le \texttt{n} < \texttt{2^(x+1)}$ where `n` is constant and `x` is a variable]. The problems you to describe a possible invariant using a specified technique. For your answer, for each candidate invariant, fill in the ??? in {???} ???**;** {**inv** ???} **while** ??? by giving an initial precondition, some initialization code, the candidate invariant, and the loop test. If you can, make an educated guess and include a range limitation on the new variable. If you can't find an initial precondition or initialization code for some case, explain why. Feel free to write `2^x` as 2^x if you prefer.

4. [15 = 5 * 3 points] There are five possible invariants when replacing a constant by a variable in q, but two of them involve replacing the `2` in `2^`… and don't lead to reasonable invariants. Describe the remaining three possibilities.

5. [6 = 2*3 points] Describe the two possible invariants when you take the $\texttt{2^x} \le \texttt{n} < \texttt{2^(x+1)}$ part of q and drop a conjunct.

6. [16 = 2 * 8 points] For each triple below, give a full proof outline for partial correctness by expanding the partial outline using *wp*'s. (Logically simplify as you calculate each *wp*.) Give definitions for the initial precondition p and the intermediate condition, logically simplified.
   a. $\{p\}$ `b[j] := a; b[i] := c` $\{\texttt{b[j]} \le \texttt{b[i]}\}$
   b. $\{p\}$ `b[j] := b[m]; b[m] := b[n]` $\{\texttt{b[j]} < \texttt{b[n]}\}$

7. [12 points] Is $\{\texttt{j} = \texttt{b[j]} < \texttt{b[k]} \le \texttt{b[b[k]]}\}$ `b[b[j]] := b[k]` $\{\texttt{b[j]} \le \texttt{b[b[k]]}\}$
   of the *wp* (which, hint: is messy).

Solution to Homework 5 — Convergence; Finding Invariants; Array Assignments

Part 1 (Finding Invariants)

1. (Total correctness of calculation)

   For the true branch to be totally correct, we need $p_0 \wedge \texttt{b[j]} \geq 0$ to $\Rightarrow D(\texttt{sqrt(b[j])}) \Leftrightarrow (0 \leq \texttt{j} < \texttt{size(b)} \wedge \texttt{b[j]} \geq 0)$. So $p_0 \Rightarrow 0 \leq \texttt{j} < \texttt{size(b)}$. For the false branch, we need $p_0 \wedge \texttt{b[j]} < 0 \Rightarrow \texttt{x} = \texttt{sqrt(b[j])}$. Since $\texttt{x} = \texttt{sqrt(b[j])}$ is impossible when $\texttt{b[j]} < 0$, we need $p_0 \wedge \texttt{b[j]} < 0 \Rightarrow$ False. I.e., we need $p_0 \Rightarrow \texttt{b[j]} \geq 0$. Altogether then, we need $p_0 \Leftrightarrow 0 \leq \texttt{j} < \texttt{size(b)} \wedge \texttt{b[j]} \geq 0$. An outline is

   $\{p_0 \equiv 0 \leq \texttt{j} < \texttt{size(b)} \wedge \texttt{b[j]} \geq 0\}$

   if b[j] ≥ 0 **then**

   $\{p_0 \wedge \texttt{b[j]} \geq 0\}$

   $\{D(\texttt{sqrt(b[j])}) \wedge \texttt{sqrt(b[j])} = \texttt{sqrt(b[j])}\}$ $\texttt{x} := \texttt{sqrt(b[j])}$ $\{\texttt{x} = \texttt{sqrt(b[j])}\}$

   else

   $\{p_0 \wedge \texttt{b[j]} < 0\}$ $\{\texttt{F}\}$ $\{\texttt{x} = \texttt{sqrt(b[j])}\}$ **skip** $\{\texttt{x} = \texttt{sqrt(b[j])}\}$

   fi $\{\texttt{x} = \texttt{sqrt(b[j])}\}$

   (Above, we have $\texttt{F} \rightarrow \texttt{x} = \texttt{sqrt(b[j])}$ before **skip**, so we're using precondition strengthening. You can instead have the implication after the **skip** and use postcondition weakening; doesn't make any difference.) Note the **else skip** is dead code (never gets executed) and the **if** test is redundant, so the outline could be simplified to $\{p_0\}$ $\texttt{x} := \texttt{sqrt(b[j])}$ $\{\texttt{x} = \texttt{sqrt(b[j])}\}$

2. (Loop bound)

   a. `n` is invalid as loop bound: It doesn't get decreased (since it's a constant). It's nonnegative.

   b. `n-j` is invalid : It can be negative. It's decreased by the loop body.

   c. `n-j+C` is valid: The invariant implies $0 < \texttt{j} \leq \texttt{n+C}$, so $\texttt{n+C-j} > 0$, and incrementing `j` decreases it.

   d. `n+j+C` is invalid: Incrementing `j` increases, not decreases it. It's nonnegative.

   e. `n-j+2*C` is valid: Since $\texttt{C} > 0$, we know $\texttt{n-j+2*C} > \texttt{n-j+C}$, which is nonnegative from part (c). Also, incrementing `j` decreases `n-j+2*C`.

3. (integer $\log_2$)

   $\{0 \leq \texttt{j} < \texttt{size(b)} \wedge \texttt{b[j]} \geq 1\}$ // D(b[j]) and $\texttt{b[j]} \geq 1$

   $\{1 = 2\text{^}0 \leq \texttt{b[j]} \wedge 0 \leq \texttt{j} < \texttt{size(b)}\}$ `x:= 1;`

   $\{\texttt{x} = 2\text{^}0 \leq \texttt{b[j]} \wedge 0 \leq \texttt{j} < \texttt{size(b)}\}$ `k:= 0;`

   $\{$**inv** $p \equiv \texttt{x} = 2\text{^}\texttt{k} \leq \texttt{b[j]} \wedge 0 \leq \texttt{j} < \texttt{size(b)}\}$

   $\{$**bd** b[j]-x$\}$ // some other bounds: b[j]-k, *ceil*(*log*$_2$(b[j]))-k,

   while $2\texttt{*x} \leq \texttt{b[j]}$ **do**

   $\{p \wedge 2\texttt{*x} \leq \texttt{b[j]} \wedge \texttt{b[j]-x} = t_0\}$ // invariant ∧ loop test ∧ bound function value

   $\{p[2\texttt{*x/x}][\texttt{k+1/k}] \wedge \texttt{b[j]-2*x} < t_0\}$ // *wp* of next statement

   `k:= k+1`

   $\{p[2\texttt{*x/x}] \wedge \texttt{b[j]-2*x} < t_0\}$ // *wp* of next statement

$\texttt{x := 2*x}$

$\{p \wedge \texttt{b[j]-x} < \texttt{t}_0\}$ // invariant and bound function has decreased

od

$\{p \wedge \texttt{2*x} > \texttt{b[j]}\}$ // invariant and negation of loop test

$\{\texttt{x} = \texttt{2\^{}k} \le \texttt{b[j]} < \texttt{2\^{}(k+1)}\}$ // our desired postcondition

Substitutions:

- $p[\texttt{2*x/x}] \equiv \texttt{2*x} = \texttt{2\^{}k} \le \texttt{b[j]} \wedge 0 \le \texttt{j} < \texttt{size(b)}$
- $p[\texttt{2*x/x}][\texttt{k+1/k}] \equiv \texttt{2*x} = \texttt{2\^{}(k+1)} \le \texttt{b[j]} \wedge 0 \le \texttt{j} < \texttt{size(b)}$

Proof Obligations:

- $\texttt{x} = \texttt{2\^{}0} \le \texttt{b[j]} \rightarrow \texttt{x} = \texttt{2\^{}n} \le \texttt{b[j]} \wedge q$
- $p \wedge \texttt{2*x} \le \texttt{b[j]} \wedge \texttt{b[j]-x} = \texttt{t}_0 \rightarrow p[\texttt{2*x/x}][\texttt{n+1/n}] \wedge \texttt{b[j]-2*x} < \texttt{t}_0$
- $p \wedge \texttt{2*x} > \texttt{b[j]} \rightarrow \texttt{x} = \texttt{2\^{}n} \le \texttt{b[j]} < \texttt{2\^{}(n+1)}$

Note: The outline above calculates the *wp* of the loop body and its postcondition. You can certainly write an alternative outline that shows that the *sp* of the loop body and its precondition implies the loop body postcondition.

4. (Replace constant by a variable) I reused variable `v` throughout, but it's fine to use other variables. The ranges (`v ≥ 0` or `1`) are just educated guesses. There may be other initializations possible.

#1 $\{\texttt{n} \ge 1\}$ `v := 0; x := ???; z := ???`

$\{$**inv** $\texttt{v} \ge 0 \wedge \texttt{x} \ge \texttt{v} \wedge \texttt{z} = \texttt{2\^{}x} \le \texttt{n} < \texttt{2\^{}(x+1)}\}$ **while** $\texttt{v} \ne 0$ …

There isn't any reasonable way to initialize `x` such that $\texttt{2\^{}x} \le \texttt{n} < \texttt{2\^{}(x+1)}$, since it is the problem we're trying to solve.

#2 $\{\texttt{n} \ge 1\}$ `v := 1; x := 0; z := 1;`

$\{$**inv** $\texttt{v} \ge 0 \wedge \texttt{x} \ge 0 \wedge \texttt{z} = \texttt{2\^{}x} \le \texttt{v} < \texttt{2\^{}(x+1)}\}$ **while** $\texttt{v} \ne \texttt{n}$ …

#3 $\{\texttt{n} \ge 1\}$ `v := n+1; x := 0; z := 1;` // `v` has to be large enough to get $\texttt{n} < \texttt{v\^{}(0+1)}$

$\{$**inv** $\texttt{v} \ge 0 \wedge \texttt{x} \ge 0 \wedge \texttt{z} = \texttt{2\^{}x} \le \texttt{n} < \texttt{v\^{}(x+1)}\}$ **while** $\texttt{v} \ne 2$ …

You weren't asked for them, but if you're interested in the two possibilities we omitted, here they are:

- $\{\texttt{n} \ge 1\}$ `v := 0; x := n; z := 1;` // `x` has to be large enough so $\texttt{n} < \texttt{2\^{}(x+1)}$

  $\{$**inv** $\texttt{v} \ge 0 \wedge \texttt{x} \ge 0 \wedge \texttt{z} = \texttt{v\^{}x} \le \texttt{n} < \texttt{2\^{}(x+1)}\}$ **while** $\texttt{v} \ne 2$ …

- $\{n \ge 1\}$ v := n; x := 0; z := 1; // v has to be large enough to get $n < 2\text{^}(0+v)$
  $\{\textbf{inv } v \ge 1 \wedge x \ge 0 \wedge z = 2\text{^}x \le n < 2\text{^}(x+v)\}$ **while** $v \ne 1$ …

5. (Drop a conjunct) Given $q \equiv x \le 0 \wedge z = 2\text{^}x \le n < 2\text{^}(x+1)$

#1 Dropping the right conjunct $n < 2\text{^}(x+1)$ makes initialization easy:

$\{n \ge 2\}$ x := 0; z := 1; $\{\textbf{inv } x \ge 0 \wedge z = 2\text{^}x \le n\}$ **while** $n \ge 2\text{^}(x+1)$ …

#2 If we drop $2\text{^}x \le n$, initialization of z is nontrivial: We need x large enough for $2\text{^}x > n \ge 1$; setting $x = n$ works, but setting $z = 2\text{^}n$ requires a loop or an exponentiation function, which we're presumably trying to avoid by writing this program. Allowing z := 2^n would give us

$\{n \ge 1\}$ x := n; z := 2^n; $\{\textbf{inv } x \ge 0 \wedge z = 2\text{^}x \wedge n < 2\text{^}(x+1)\}$ **while** $2\text{^}x > n$ …

6. (*wp* of array assignments)

6a. For the full proof outline, we find p and q such that $\{p\}$ b[j] := a; $\{q\}$ b[i] := c $\{b[j] \le b[i]\}$ using *wp*. The calculation steps below are shown in detail so you can see all of them, but shorter answers could be okay.

$q \equiv wp(b[i] := c, b[j] \le b[i]) \equiv (b[j] \le b[i])[c/b[i]]$

$\equiv (b[j])[c/b[i]] \le (b[i])[c/b[i]]$

$\equiv \textbf{if } j = i \textbf{ then } c \textbf{ else } b[j] \textbf{ fi} \le c$

$\Leftrightarrow j = i \vee b[j] \le c$ (or anything equivalent, like $j \ne i \rightarrow b[j] \le c$)

$p \equiv wp(b[j] := a, q)$

$\equiv q[a/b[j]]$

$\equiv (j = i \vee b[j] \le c)[a/b[j]]$

$\Leftrightarrow j = i \vee a \le c$ (or anything equivalent, like $j \ne i \rightarrow a \le c$)

6b. Again, let's use *wp* to find p and q in $\{p\}$ b[j] := b[m]; $\{q\}$ b[m] := b[n] $\{b[j] < b[n]\}$

$q \Leftrightarrow wp(b[m] := b[n], b[j] < b[n])$

$\equiv (b[j] < b[n])[b[n]/b[m]]$

$\equiv (b[j])[b[n]/b[m]] < (b[n])[b[n]/b[m]]$

$\equiv \textbf{if } j = m \textbf{ then } b[n] \textbf{ else } b[j] \textbf{ fi}$

$< \textbf{if } n = m \textbf{ then } b[n] \textbf{ else } b[n] \textbf{ fi}$

$\Leftrightarrow \textbf{if } j = m \textbf{ then } b[n] \textbf{ else } b[j] \textbf{ fi} < b[n]$

$\Leftrightarrow j \ne m \wedge b[j] < b[n]$

$p \Leftrightarrow wp(b[j] := b[m], q)$

$\equiv q[b[m]/b[j]]$

$\equiv (j \ne m \wedge b[j] < b[n])[b[m]/b[j]]$

$\equiv \texttt{j} \neq \texttt{m} \wedge (\texttt{b[j]})[\,\texttt{b[m]}\,/\,\texttt{b[j]}\,] < (\texttt{b[n]})[\,\texttt{b[m]}\,/\,\texttt{b[j]}\,]$

$\equiv \texttt{j} \neq \texttt{m} \wedge \texttt{b[m]} < \textbf{if } \texttt{n} = \texttt{j} \textbf{ then } \texttt{b[m]} \textbf{ else } \texttt{b[n]} \textbf{ fi}$

$\Leftrightarrow \texttt{j} \neq \texttt{m} \wedge \texttt{n} \neq \texttt{j} \wedge \texttt{b[m]} < \texttt{b[n]}$

7. Let $p \equiv \texttt{j} = \texttt{b[j]} < \texttt{b[k]} \leq \texttt{b[b[k]]}$, $S \equiv \texttt{b[b[j]]} := \texttt{b[k]}$, and $q \equiv \texttt{b[j]} \leq \texttt{b[b[k]]}$. To check for validity of $\{p\}\ S\ \{q\}$, we'll see if $p \Rightarrow wp(S, q)$. Since S is an assignment, we need to substitute its right-hand side for its left-hand side in both $\texttt{b[j]}$ and $\texttt{b[b[k]]}$ (the basic parts of q). Let's calculate the two substitutions separately. Recall that need $(expr)[rhs\,/\,lhs]$ where $rhs \equiv \texttt{b[k]}$ and $lhs \equiv \texttt{b[b[j]]}$.

- First, $(\,\texttt{b[j]}\,)[\,\texttt{b[k]}\,/\texttt{b[b[j]]}\,] \equiv \textbf{if } \texttt{j} = \texttt{b[j]} \textbf{ then } \texttt{b[k]} \textbf{ else } \texttt{b[j]} \textbf{ fi}$

- To calculate $(\,\texttt{b[b[k]]}\,)[\,\texttt{b[k]}\,/\texttt{b[b[j]]}\,]$, we first need to substitute into the index $\texttt{b[k]}$: Let $e' \equiv (\,\texttt{b[k]}\,)[\,\texttt{b[k]}\,/\texttt{b[b[j]]}\,] \equiv \textbf{if } \texttt{k} = \texttt{b[j]} \textbf{ then } \texttt{b[k]} \textbf{ else } \texttt{b[k]} \textbf{ fi}$, which reduces to simply $\texttt{b[k]}$, then

  $(\,\texttt{b[b[k]]}\,)[\,\texttt{b[k]}\,/\texttt{b[b[j]]}\,]$

  $\equiv \textbf{if } e' = \texttt{b[j]} \textbf{ then } \texttt{b[k]} \textbf{ else } \texttt{b[}e'\texttt{]} \textbf{ fi}$

  $\equiv \textbf{if } \texttt{b[k]} = \texttt{b[j]} \textbf{ then } \texttt{b[k]} \textbf{ else } \texttt{b[b[k]]} \textbf{ fi}$ // $e' = \texttt{b[k]}$ from above

- Going back to the original problem,

  $wp(\,\texttt{b[b[j]]} := \texttt{b[k]},\ \texttt{b[j]} \leq \texttt{b[b[k]]}\,)$

  $\equiv (\,\texttt{b[j]}\,)[\,\texttt{b[k]}\,/\texttt{b[b[j]]}\,] \leq (\,\texttt{b[b[k]]}\,)[\,\texttt{b[k]}\,/\texttt{b[b[j]]}\,]$

  $= \textbf{if } \texttt{j} = \texttt{b[j]} \textbf{ then } \texttt{b[k]} \textbf{ else } \texttt{b[j]} \textbf{ fi}$

  $\quad \leq \textbf{if } \texttt{b[k]} = \texttt{b[j]} \textbf{ then } \texttt{b[k]} \textbf{ else } \texttt{b[b[k]]} \textbf{ fi}$

- For the triple $\{p\}\ S\ \{q\}$ to be valid, we need $p \Rightarrow wp(S, q)$ where $p \equiv \texttt{j} = \texttt{b[j]} < \texttt{b[k]} \leq \texttt{b[b[k]]}$, and $wp(S, q)$ is what we just calculated. Assume $\texttt{j} = \texttt{b[j]} < \texttt{b[k]} \leq \texttt{b[b[k]}$, then we can reduce the wp:

  $\textbf{if } \texttt{j} = \texttt{b[j]} \textbf{ then } \texttt{b[k]} \textbf{ else } \texttt{b[j]} \textbf{ fi}$

  $\quad \leq \textbf{if } \texttt{b[k]} = \texttt{b[j]} \textbf{ then } \texttt{b[k]} \textbf{ else } \texttt{b[b[k]]} \textbf{ fi}$

  $\Leftrightarrow \texttt{b[k]} \leq \textbf{if } \texttt{b[k]} = \texttt{b[j]} \textbf{ then } \texttt{b[k]} \textbf{ else } \texttt{b[b[k]]} \textbf{ fi}$ Because $\texttt{j} = \texttt{b[j]}$

  $\Leftrightarrow \texttt{b[k]} \leq \texttt{b[b[k]]}$ Because $\texttt{b[j]} < \texttt{b[k]}$, not $= \texttt{b[k]}$

  $\Leftrightarrow$ T Because $\texttt{b[k]} \leq \texttt{b[b[k]}$

