Proof Outlines; Total Correctness

CS 536: Science of Programming, Spring 2023

Due Wed Apr 5 ~~Mon Apr 3~~, 11:59 pm

2023-04-03 p.3, 2023-04-14: pp. 1, 3, 4.

A. Why?

- A formal proof lets us write out in detail the reasons for believing that something is valid.
- Proof outlines condense the same information as a proof.
- Total correctness takes correct results and adds avoidance of runtime errors and divergence.

B. Outcomes

- After this homework, you should be able to
- Translate between full proof outlines and formal proofs of partial correctness.
- Translate between a full proof outline and a minimal proof outline.
- Check an outline for convergence and avoidance of runtime errors.

C. Problems [60 points total]

Classes 16 &17: Proof Outlines [25 points]

1. [15 points] Show the full outline derived from the full proof.

1.	$\{ n > 0 \}\ k := n–1\ \{ n > 0 \wedge k = n–1 \}$	assignment (fwd)
2.	$\{ n > 0 \wedge k = n–1 \}\ x := n\ \{ n > 0 \wedge k = n–1 \wedge x = n \}$	assignment (fwd)
3.	$n > 0 \wedge k = n–1 \wedge x = n \rightarrow p$ (where $p \equiv 1 \le k \le n \wedge x = n!/k!$)	predicate logic
4.	$\{ n > 0 \wedge k = n–1 \}\ x := n\ \{ p \}$	postcond. weak. 2, 3
5.	$\{ n > 0 \}\ k := n–1\ ;\ x := n\ \{ p \}$	sequence 1, 4
6.	$\{ p[x*k/x] \}\ x := x*k\ \{ p \}$	assignment (bwd)
7.	$\{ p[x*k/x][k–1/k] \}\ k := k–1\ \{ p[x*k/x] \}$	assignment (bwd)
8.	$p \wedge k > 1 \rightarrow p[x*k/x][k–1/k]$	predicate logic
9.	$\{ p \wedge k > 1 \}\ k := k–1\ \{ p[x*k/x] \}$	precondition str. 8, 7
10.	$\{ p \wedge k > 1 \}\ k := k–1\ ;\ x := x*k\ \{ p \}$	sequence 9, 6
11.	$\{$ ***inv*** $p \}\ W\ \{ p \wedge k \le 1 \}$ // where $W \equiv$ ***while*** $k > 1$ ***do*** $k := k–1\ ;\ x := x*k$ ***od***	while loop 10
12.	$\{ n > 0 \}\ k := n–1\ ;\ x := n\ \{$ ***inv*** $p \}\ W\ \{ p \wedge k \le 1 \}$	sequence 5, 11
13.	$p \wedge k \le 1 \rightarrow x = n!$	predicate logic [2023-04-14]
14.	$\{ n > 0 \}\ k := n–1\ ;\ x := n\ \{$ ***inv*** $p \}\ W\ \{ x = n! \}$	postcond. weak. 12, 13

Expanded substitutions: (You don't have to re-include this with your outline)

- $p \equiv 1 \le k \le n \wedge x = n!/k!$

- $p\,[\,x*k/x\,] \equiv 1 \le k \le n \wedge x*k = n!/k!$
- $p\,[\,x*k/x\,]\,[\,k{-}1/k\,] \equiv 1 \le k-1 \le n \wedge x*(k-1) = n!/(k-1)!$

2. [10 points] Give a full proof outline obtained by expansion of the partial proof outline below. Work backward though the program (use *wp* on the four assignments). Show the results of substitutions somewhere.

$\{ y \ge 1 \}\ x := 0\ ;\ r := 1\ ;$
$\{ \textbf{inv}\ \ p \equiv 1 \le r = 2\,\hat{}\,x \le y \}$
while $2*r \le y$ ***do***
  $r := 2*r\ ;\ x := x+1$
od
$\{ r = 2\,\hat{}\,x \le y \le 2\,\hat{}\,(x+1) \}$

Class 18: Total Correctness: Errors and Divergence [35 points total]

Convergence [14 points]

3. [6 points] For $\{ \textbf{inv}\ p \}\,\{ \textbf{bd}\ t \}$ ***while*** B ***do*** S ***od*** $\{ p \wedge \neg B \}$, for each of the following properties, in order to get convergence, must the property hold? If not, can it hold? Must it never hold? Briefly discuss each answer.
   a. $\{ p \wedge B \wedge t > t_0 \}\ S\ \{ t = t_0 \}$
   b. $p \wedge t = 0 \to \neg B$
   c. $p \wedge t > 0 \to B$
   d. $p \wedge \neg B \to t = 0$
   e. $(p \wedge B \wedge t = t_0) \to wp(S, t < t_0)$
   f. $sp(p \wedge B \wedge t = t_0, S) \to t < t_0$

4. [8 = 4 * 2 points] Consider the loop $\{ \textbf{inv}\ p \}\,\{ \textbf{bd}\ t \}$ ***while*** $k \le n$ ***do*** ... $k := k+1$ ***od***.
   Assume $p \to (n \ge 0 \wedge 0 < C \le k \le n+C)$ where C is a named constant, not necessarily ≥ 0. 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. $n-k$
   b. $n-k+C$
   c. $n+k+C$
   d. $2\,\hat{}\,(n+C)/2\,\hat{}\,k$

Runtime Errors and Convergence [21 points]

5. [21=7*3 points] The program below is outlined for partial correctness, with initial values given for the predicates and for the bound function t. Rewrite the outline for total correctness. This will entail a number of steps:

a. Fix t (Hint: the initial value is too small). Give your new t as the answer to this part.

b. Fix p to make it safe: Calculate $D(p)$ and redefine p as the old $p \wedge D(p)$. Give $D(p)$ and the new p.

c. Fix p_0: Make it safe and make $p_0 \wedge k{=}1$ imply p. Give the new p_0.

d. Verify that p_3 is safe: Calculate $D(p_3)$ and make sure $p_3 \Rightarrow D(p)$. If it isn't, modify p_3 (i.e., modify p and/or t) and go back to (a) or (b) as necessary. Give $D(p_3)$.

e. Calculate p_2 as the wp of the loop body and p_3, then verify that p_2 is safe and that $p_1 \Rightarrow p_2$. If not, fix p_1 or p_2 as appropriate. Give p_2.

f. Fix q by making it safe: Calculate $D(q)$ and redefine q as the old $q \wedge D(q)$. We should have $p_4 \Rightarrow q$. If not, something's wrong with p_4 or p or q. Give $D(q)$ and the new q.

g. Let $B \equiv sqrt(k) < x/y$ (the loop test). Calculate $D(B)$ and verify that B doesn't imply $D(B)$. We could modify the program to use ***while*** $\downarrow B$, but do we need to? If not, explain why, briefly.

In the program below, the definitions of p_0, p, t, and q will change (which is why they're "initial"). The definitions of p_1, p_2, and p_3 will change only because p and t change.

$\{p_0\}$ // initial $p_0 \equiv T$
$k := 1$;
$\{p_0 \wedge k = 1\}$
$\{$ ***inv*** $p\}$ // initial $p \equiv sqrt(k{-}1) \le x/y$ [2023-04-14: < is correct]*
$\{$ ***bd*** $t\}$ // initial $t \equiv (x/y)^2 - k$ [$t \equiv (x/y)^2 - (k-1)$ is the final t. Explain why $(x/y)^2 - k$ is bad
while $sqrt(k) < x/y$ ***do*** and $(x/y)^2 - (k-1)$ is right]
  $\{p_1\}$ // $p_1 \equiv p \wedge sqrt(k) < x/y \wedge t = t_0$
  $\{p_2\}$ // $p_2 \equiv wp(k := k+1, p_3)$
  $k := k+1$
  $\{p_3\}$ // $p_3 \equiv p \wedge t < t_0$
od
$\{p_4 \equiv p \wedge \neg test\}$ [2023-04-14]
$\{q\}$ // initial $q \equiv sqrt(k{-}1) < x/y \le sqrt(k)$

[2023-04-14] To get p, we dropped the second conjunct of $q \equiv sqrt(k-1) < x/y \le sqrt(k)$. So p is the part of q that's left, namely $sqrt(k-1) < x/y$, and the while test is $\neg(x/y \le sqrt(k))$.

Solution to Homework 8

Classes 16 & 17: Proof Outlines

1. (Full outline from formal proof)

   $\{ n > 0 \}$
   $k := n–1 ; \{ n > 0 \wedge k = n–1 \}$
   $x := n ; \{ n > 0 \wedge k = n–1 \wedge x = n \}$
   $\{ \textbf{inv}\ p \}\ \textbf{while}\ k > 1\ \textbf{do}$ // where $p \equiv 1 \le k \le n \wedge x = n!/k!$
   $\quad \{ p \wedge k > 1 \}$
   $\quad \{ p[x*k/x][k–1/k] \}\ k := k–1 ;$
   $\quad \{ p[x*k/x] \}\ x := x*k$
   $\quad \{ p \}$
   od
   $\{ p \wedge k \le 1 \}$
   $\{ x = n! \}$ ⇐ *to get this line we needed lines 13 and 14 in the formal proof. [2023-04-14]*

2. (Expand partial outline)

   $\{ y \ge 1 \}$
   $\{ p[1/r][0/x] \}\ x := 0 ;$ // $p[1/r][0/x] \equiv 1 \le 1 = 2^0 \le y$
   $\{ p[1/r] \}\ r := 1 ;$ // $p[1/r] \equiv 1 \le 1 = 2^x \le y$
   $\{ \textbf{inv}\ p \equiv 1 \le r = 2^x \le y \}$
   while $2*r \le y$ **do**
   $\quad \{ p \wedge 2*r \le y \}$
   $\quad \{ p[x+1/x][2*r/r] \}\ r := 2*r ;$ // $p[x+1/x][2*r/r] \equiv 1 \le 2*r = 2^(x+1) \le y$
   $\quad \{ p[x+1/x] \}\ x := x+1$ // $p[x+1/x] \equiv 1 \le r = 2^(x+1) \le y$
   $\quad \{ p \}$
   od
   $\{ p \wedge 2*r > y \}$
   $\{ r = 2^x \le y \le 2^(x+1) \}$

Class 18: Total Correctness

3. (Convergence of $\{ \textbf{inv}\ p \}\ \{ \textbf{bd}\ t \}\ \textbf{while}\ B\ \textbf{do}\ S\ \textbf{od}\ \{ p \wedge \neg B \}$)
   a. Must be true: $\{ p \wedge B \wedge t > t_0 \}\ S\ \{ t = t_0 \}$. Whatever t is at the end of the iteration; it needed to be larger at the start of the iteration.
   b. Must be true: $p \wedge t = 0 \rightarrow \neg B$. If $t = 0$ at the start of an iteration, decreasing it would make t negative at the end of the iteration.
   c. Can be false: $p \wedge t > 0 \rightarrow B$. We can have $t > 0$ on loop termination.
   d. Can be false: $p \wedge \neg B \rightarrow t = 0$. Again, $t > 0$ at loop termination is allowed.

e. Must be true: $(p \wedge B \wedge t = t_0) \rightarrow wp(S, t < t_0)$. This guarantees that S reduces t.

f. Must be true: $sp(p \wedge B \wedge t = t_0, S) \rightarrow t < t_0$. This also guarantees that S reduces t.

4. (Possible bound functions for $\{$ ***inv*** $p\}$ $\{$ ***bd*** $t\}$ ***while*** $k \le n$ ***do*** ... $k := k+1$ ***od***, where we have $p \rightarrow (n \ge 0 \wedge 0 < C \le k \le n + C$, for constant C (which can be <, =, or > 0).

a. $(n–k)$: Is decreased by incrementing k, but it can't be a bound function because it can be negative. Since $k \le n + C$, we can subtract $C+k$ from both sides and get $k – (C+k) \le n + C – (C+k)$, which simplifies to $–C \le n–k$.

b. $n–k+C$: Can be a bound function. Since $k \le n + C$, we know $0 \le n–k+C$, so it's nonnegative, and incrementing k decreases $n–k+C$.

c. $n+k+C$: Cannot be a bound function because increasing k makes $n+k+C$ larger, not smaller. (It's nonnegative, however: $0 < C \le k \le n+C \Rightarrow 0 < n+C \Rightarrow k < n+k+C$.)

d. $2 \wedge (n+C)/2 \wedge k$: Can be a bound function. It's decreased by incrementing k, and it's nonnegative because $0 \le k \le n+C \Rightarrow 2 \wedge k \le 2 \wedge (n+C) \Rightarrow 2 \wedge (n+C)/2 \wedge k \ge 1$.

5. (Runtime errors and convergence)

a. The problem is that $(x/y)^2 – k$ is negative if $x/y = 0$ and $k=1$. Change t by adding 1 so that now, $t \equiv (x/y)^2 – k+1$.[1]

b. If $p \equiv sqrt(k–1) < x/y$ then $D(p) \Leftrightarrow y \ne 0 \wedge k \ge 1$. Redefine $p \equiv y \ne 0 \wedge k \ge 1 \wedge sqrt(k–1) < x/y$

c. Change p_0 to $y \ne 0$ so that $p_0 \wedge k = 1 \Rightarrow p$: I.e., $(y \ne 0 \wedge k = 1) \Rightarrow y \ne 0 \wedge k \ge 1 \wedge sqrt(k–1) < x/y$.

d. We calculate $p_3 \equiv p \wedge t < t_0 \equiv (y \ne 0 \wedge k \ge 1 \wedge sqrt(k–1) < x/y) \wedge ((x/y)^2 – k+1) < t_0$. Since $D(p_3) \Leftrightarrow k \ge 1 \wedge y \ne 0$ and $p_3 \Rightarrow D(p_3)$, p_3 is safe.

e. $p_2 \equiv wp(k := k+1, p_3) \equiv p_3[k+1/k] \equiv (y \ne 0 \wedge k + 1 \ge 1 \wedge sqrt(k+1–1) < x/y) \wedge ((x/y)^2 – (k+1)+1) < t_0$. Since $D(p_2) \Leftrightarrow$ [2023-04-14] $y \ne 0 \wedge k \ge 0$ and $p_2 \Rightarrow D(p_2)$, so p_2 is safe.

f. With $q \equiv sqrt(k–1) < x/y \le sqrt(k)$, We have $D(q) \Leftrightarrow k \ge 1 \wedge y \ne 0$, but q doesn't imply $D(q)$, so we'll redefine q to be the old $(q \wedge D(q))$, which makes the new q safe. The implication $p_4 \Rightarrow q$ does hold, so the predicate logic obligation is met.

(If you want details for $p_4 \Rightarrow q$, we have $p_4 \equiv p \wedge sqrt(k) \ge x/y$ and $q \equiv sqrt(k–1) < x/y \le sqrt(k) \wedge k \ge 1 \wedge y \ne 0$. (1) Most of q holds because $p_4 \Rightarrow p$, and p includes $k \ge 1$, $y \ne 0$, and $sqrt(k–1) < x/y$. (2) The remainder of q is $x/y \le sqrt(k)$, which is included in p_4.)

g. The loop test $B \equiv sqrt(k) < x/y$, so $D(B) \Leftrightarrow k \ge 0 \wedge y \ne 0$, and B doesn't imply $D(B)$. This makes $\downarrow B \Leftrightarrow k \ge 0 \wedge y \ne 0 \wedge sqrt(k) < x/y$. We could change the program to use ***while*** $\downarrow B$, but the invariant implies $k \ge 0 \wedge y \ne 0$, so adding it to the loop test is redundant.

---

[1] Just a side mention: We can't use $x/y – sqrt(k)$ as a bound function because it's not always reduced by incrementing k (because of truncation). E.g., $sqrt(4) = sqrt(5) = 2$.

