Proof Outlines; Convergence

CS 536: Science of Programming, Spring 2021

Due Fri Apr 9, 11:59 pm

4/7 p.2

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.

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.

C. Problems [50 points total]

Class 17: Full Proof Outlines [10 points]

1. [10 pts] Give the full proof outline that corresponds to the formal proof below.

   1. \{n > 0\} k := n–1 \{n > 0 \land k = n–1\} Assignment
   2. \{n > 0 \land k = n–1\} p := n \{n > 0 \land k = n–1 \land p = n\} Assignment
   3. \{n > 0\} k := n–1; p := n \{n > 0 \land k = n–1 \land p = n\} Sequence 1, 2
   4. n > 0 \land k = n–1 \land p = n \rightarrow p Predicate logic
      where p = 1 \leq k \leq n \land p = n!/k!
   5. \{n > 0\} k := n–1; p := n \{p\} Postcond. weak 3, 4
   6. \{p[p/k/p]\} p := p*k \{p\} Assignment
   7. \{p[p/k/p][k–1/k]\} k := k–1 \{p[p/k/p]\} Assignment
   8. p \land k > 1 \rightarrow p[p/k/k][k–1/k] Predicate logic
   9. \{p \land k > 1\} k := k–1 \{p[p/k/k]\} Precond. str 8, 7
   10. \{p \land k > 1\} k := k–1; p := p*k \{p\} Sequence 9, 6
   11. \{inv p\} W \{p \land k \leq 1\} While 10
       where W = while k > 1 do k := k–1; p := p*k od
   12. p \land k \leq 1 \rightarrow p = n! Predicate logic
   13. \{inv p\} W \{p = n!\} Postcond. weak 12, 11
   14. \{n > 0\} k := n–1; p := n; \{inv p\} W \{p = n!\} Sequence 5, 13
Class 17: Partial Proof Outlines [20 points total]

4. [10 pts] Give a full proof outline obtained by expansion of the partial proof outline below.
   Work forward through the program (use sp on the four assignments and if-else statement).
   Feel free to use substitution notation (predicate[expr/var]) but show their expansions somewhere.
   \[
   \{q = r = X*Y - x*y\}
   \text{if } \text{even}(x) \text{ then}
   \begin{align*}
   y & : = 2*y; x := x/2 \\
   \text{else}
   r & : = r + y; x := x - 1
   \end{align*}
   \text{fi } \{q\}
   \]

5. [10 pts] 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.
   \[
   \{y \geq 1\} x := 0; r := 1;
   \{\text{inv } p = 1 \leq r = 2^x \leq y\}
   \text{while } 2*r \leq y \text{ do}
   \begin{align*}
   r & : = 2*r; x := x + 1
   \end{align*}
   \text{od}
   \{r = 2^x \leq y \leq 2^{(x+1)}\}
   \]

Class 18: Convergence [20 points total]

1. [10 = 5 * 2 points] Consider the loop
   \[
   \{\text{inv } p\} \{\text{bd } t\} \text{ while } k \leq n \text{ do } k := k+1 \text{ od}
   \]
   Assume \( p \rightarrow (n \geq 0 \land 0 < C \leq k \leq n+C) \) (where \( C \) is a named constant). 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). Include a list of predicate logic obligations and show the expansion of any substitutions.
   a. \( n - k \)
   b. \( n + k + C \)
   c. \( n - k + C \) [4/7 removed typo]
   d. \( n - k + 2*C \)
   e. \( 2^{n+C}/2^k \)
2. [10 points] Complete the proof of total correctness of the program below by filling in the missing pieces that ensure convergence. If you want, feel free to define other predicates ("Let name = predicate it stands for.") Give a list of predicate logic obligations and the results of any substitutions (predicate[expr/var] = ???)

```
{??? ∧ 0 ≤ c < |b|}² // (Add an initial precondition to ensure bound fcn ≥ 0)
x := 1;
{??}
// (Add sp of the assignment above.)
k := 0;
{??}
// (Add sp of the assignment above.)
{inv p = x = 2^k ≤ b[c]} ∧ 0 ≤ c < |b| ∧ ??} // (Augment p to ensure bound expression ≥ 0)
{bd ??}
// (Insert bound expr. Hint: has to include k and/or x.)
while 2*x ≤ b[c] do
  {p ∧ 2*x ≤ b[c] ∧ ???} // (Add bound expr. = logical variable.)
  {??}
  // (Add wp of the assignment below.)
  k := k+1;
  {??}
  // (Add wp of the assignment below.)
  x := 2*x
  {p ∧ ???} // (Add bound expr < logical variable.)
od
{p ∧ 2*x > b[c]}
{x = 2^k ≤ b[c] < 2^(k+1)}
```

---

1. For the other part of total correctness, \(D(b[c])\) has already been added to avoid runtime errors.

2. I've used \(|b|\) as a synonym for size(b). Feel free to use either notation.