Proofs and Proof Outlines for Partial Correctness

Part 1: Full Proofs and Proof Outlines of Partial Correctness

CS 536: Science of Programming, Fall 2022

2022-10-19: p.3

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. Objectives

At the end of this class you should

- Know how to write and check a formal proof of partial correctness.
- Know how to translate between full formal proofs and full proof outlines

C. Formal Proofs of Partial Correctness

- As you've seen, the format of a formal proof is very rigid syntactically. The relationship between formal proofs and informal proofs is like the description of an algorithm in a program (very rigid syntax) versus in pseudocode (much more informal syntax).
- Just as a reminder, we're using Hilbert-style proofs: Each line's assertion is an assumption, an axiom, or follows by some rule that appeals to earlier lines in the proof. In high-school geometry, we might have used

1.	*Length of AB = length of XY*	*Assumption*
2.	*Angle ABC = Angle XYZ*	*Assumption*
3.	*Length of BC = length of YZ*	*Assumption*
4.	*Triangles ABC, XYZ are congruent*	*Side-Angle-Side, lines 1, 2, 3*

D. Sample Formal Proofs

- We can write out the reasoning for the sample summation loop we looked at. We've seen formal proofs of the loop body's correctness; all we really have to do is attach the proof of loop initialization correctness:

Example 1: *Simple summation program*

$\{n \geq 0\}$
$k := 0;\ s := 0;$
$\{$***inv*** $p_1 \equiv 0 \leq k \leq n \wedge s = sum(0, k)\}$
while $k < n$ ***do***
  $s := s+k+1;\ k := k+1$
od
$\{s = sum(0, n)\}$

- Below, let $S_1 \equiv s := s+k+1;\ k := k+1$ (the loop body) and let $W \equiv$ ***while*** $k < n$ ***do*** S_1 ***od*** (the loop).

1.	$\{n \geq 0\}\ k := 0\ \{n \geq 0 \wedge k = 0\}$	*assignment (forward)*
2.	$\{n \geq 0 \wedge k = 0\}\ s := 0\ \{n \geq 0 \wedge k = 0 \wedge s = 0\}$	*assignment*
3.	$\{n \geq 0\}\ k := 0;\ s := 0\ \{n \geq 0 \wedge k = 0 \wedge s = 0\}$	*sequence 1, 2*
4.	$n \geq 0 \wedge k = 0 \wedge s = 0 \rightarrow p_1$ *where* $p_1 \equiv 0 \leq k \leq n \wedge s = sum(0, k)$	*predicate logic*
5.	$\{n \geq 0\}\ k := 0;\ s := 0\ \{p_1\}$	*postcondition weakening, 3, 4*
6.	$\{p_1[k+1/k]\}\ k := k+1\ \{p_1\}$	*assignment (backward)*
7.	$\{p_1[k+1/k][s+k+1/s]\}\ s := s+k+1\ \{p_1[k+1/k]\}$	*assignment*
8.	$\{p_1[k+1/k][s+k+1/s]\}\ S_1\ \{p_1\}$	*sequence 7, 6*
9.	$p_1 \wedge k < n \rightarrow p_1[k+1/k][s+k+1/s]$	*predicate logic*
10.	$\{p_1 \wedge k < n\}\ S_1\ \{p_1\}$	*precondition strengthening, 9, 8*
11.	$\{$***inv*** $p_1\}$ ***while*** $k < n$ ***do*** S_1 ***od*** $\{p_1 \wedge k \geq n\}$	***while*** *loop, 10*
12.	$\{n \geq 0\}\ k := 0;\ s := 0;\ W\ \{p_1 \wedge k \geq n\}$ *(where W is the loop in line 11)*	*sequence 5, 11*
13.	$p_1 \wedge k \geq n \rightarrow s = sum(0, n)$	*predicate logic*
14.	$\{n \geq 0\}\ k := 0;\ s := 0;\ W\ \{s = sum(0, n)\}$	*postcond. weakening, 12, 13*

- The proof uses two substitutions:
  - $p_1[k+1/k] \equiv 0 \leq k+1 \leq n \wedge s = sum(0, k+1)$
  - $p_1[k+1/k][s+k+1/s] \equiv (0 \leq k \leq n \wedge s = sum(0, k+1))[s+k+1/s]$
    $\equiv 0 \leq k+1 \leq n \wedge s+k+1 = sum(0, k+1)$
- The proof also gives us three predicate logic obligations (implications we need to be true, otherwise the overall proof is incorrect). Happily, all three are in fact valid.
  - $n \geq 0 \wedge k = 0 \wedge s = 0 \rightarrow p_1$
    $\equiv n \geq 0 \wedge k = 0 \wedge s = 0 \rightarrow 0 \leq k \leq n \wedge s = sum(0, k)$
  - $p_1 \wedge k < n \rightarrow p_1[k+1/k][s+k+1/s]$
    $\equiv (0 \leq k \leq n \wedge s = sum(0, k)) \wedge k < n \rightarrow 0 \leq k+1 \leq n \wedge s+k+1 = sum(0, k+1)$
  - $p_1 \wedge k \geq n \rightarrow s = sum(0, n)$
    $\equiv (0 \leq k \leq n \wedge s = sum(0, k)) \wedge k \geq n \rightarrow s = sum(0, n)$

- To review, the order of the lines in the proof is somewhat arbitrary — you can only refer to lines above you in the proof, but they can be anywhere above you.
  - For example, lines 1 and 2 don't have to be in that order, they just have to be before we use them in the sequence rule at line 3 (which in turn has to be somewhere before line 5, and so on).

E. Full Proof Outlines

- Formal proofs are long and contain repetitive information (we keep copying the same conditions over and over). All in all, they're too tedious to use.
- A ***proof outline*** is a way to write out all the information that you would need to generate a full formal proof, but with less repetition, so they're much shorter, and they don't mask the overall structure of the program the way a full proof does.
  - To get a proof outline, we annotate program statements with their preconditions and postconditions, so that every statement in the program is part of one or correctness triples.
    - Every triple must be provable using the proof rules.
    - We include all statements, not just basic ones like assignments and ***skip***.

Proof Outlines for Individual Statements

- Each instance of a proof rule corresponds to a proof outline that combines the antecedents (if any) and consequent of the rule. (For a loop, the loop body, for conditionals, each branch.)

Assignment and skip

- These triples are annotated exactly as they are in the proof rules.
  - $\{p\}\ x := e\ \{q\}$
  - $\{p\}$ ***skip*** $\{p\}$

Sequence

- To combines $\{p_1\}\ S_1\ \{q\}$ and $\{q\}\ S_2\ \{q_1\}$ to get $\{p_1\}\ S_1;\ S_1\ \{q_1\}$, we include the condition q that sits between S_1 and S_2:
  - $\{p_1\}\ S_1\ ;\ \{q\}\ S_2\ \{q_1\}$

While loops

- There is only one loop rule hence only one triple. It combines triple for the body, $\{p \wedge B\}\ S\ \{p\}$, and the triple for the overall statement, $\{$***inv*** $p\}$ ***while*** B ***do*** S ***od*** $\{p \wedge \neg B\}$.
  - $\{$***inv*** $p\}$ ***while*** B ***do*** $\{p \wedge B\}\ S\ \{p\}$ ***od*** $\{p \wedge \neg B\}$

Conditionals

- There are multiple possibilities for conditionals because we have multiple rules for them. Each outline includes the triples for the branches and the triple for the overall conditional statement.
  - $\{p\}$ ***if*** B ***then*** $\{p \wedge B\}\ S_1\ \{q_1\}$ ***else*** $\{p \wedge \neg B\}\ S_2\ \{q_2\}$ ***fi*** $\{q_1 \vee q_2\}$ [2022-10-18]
  - $\{(B \rightarrow p_1) \wedge (\neg B \rightarrow p_2)\}$ ***if*** B ***then*** $\{p_1\}\ S_1\ \{q_1\}$ ***else*** $\{p_2\}\ S_2\ \{q_2\}$ ***fi*** $\{q_1 \vee q_2\}$ [2022-10-18]

- $\{p\}$ ***if*** $B_1 \rightarrow \{p \wedge B_1\}\ S_1\ \{q_1\} \,\square\, B_2 \rightarrow \{p \wedge B_2\}\ S_2\ \{q_2\}$ ***fi*** $\{q_1 \vee q_2\}$
- $\{(B_1 \rightarrow p_1) \wedge (B_2 \rightarrow p_2\}$ ***if*** $B_1 \rightarrow \{p_1\}\ S_1\ \{q_1\} \,\square\, B_2 \rightarrow \{p_2\}\ S_2\ \{q_2\}$ ***fi*** $\{q_1 \vee q_2\}$

Strengthening and Weakening

- For strengthening or weakening operations, we include a condition for the new condition, next to the condition it replaces:
  - $\{p_1\}\ \{p\}\ S\ \{q\}$ For strengthening using $p_1 \rightarrow p$
  - $\{p\}\ S\ \{q\}\ \{q_1\}$ For weakening using $q \rightarrow q_1$.
- Just generally in an outline, if two conditions sit next to each other, say $\{p\}\ \{q\}$, this indicates a predicate logic implication $p \rightarrow q$.

Full Outlines Aren't Unique

- A proof outline does not stand for a unique proof. (Unless you have a one-line proof.)
  - One reason is pretty trivial: If a rule has more than one antecedent, they can be shown in any order. I.e., for a conditional, the triples for the true branch and false branch can appear in that order or the reverse.
  - The other reason is that strengthening and weakening operations within a sequence aren't unique. The overall proof ends up with the same triple, but the path there might be different.
  - E.g., take $\{p_1\}\ S_1;\ \{p_2\}\ \{p_3\}\ S_2\ \{p_4\}$. We can read this as
    - $\{p_1\}\ S_1\ \{p_2\}\ \{p_3\}$, combined with $\{p_3\}\ S_2\ \{p_4\}$ or
    - $\{p_2\}\ \{p_3\}\ S_2\ \{p_4\}$ combined with $\{p_1\}\ S_1\ \{p_2\}$.
    - I.e., either we weaken the postcondition of S_1 or we strengthen the precondition of S_2.
  - Luckily, the difference is hardly ever a problem. It's often just a style issue*.

Example 1

- One kind of problem to study is "What is the full proof that corresponds to this outline?"
- E.g., what is the outline for $\{T\}\ k := 0;\ \{k = 0\}\ x := 1\ \{k = 0 \wedge x = 1\}\ \{k \geq 0 \wedge x = 2\text{^}k\}$?
- The basic structure is that we form the sequence $k := 0;\ x := 1$ and then weaken its postcondition.

1.	$\{T\}\ k := 0\ \{k = 0\}$	*assignment (forward)*
2.	$\{k = 0\}\ x := 1\ \{k = 0 \wedge x = 1\}$	*assignment (forward)*
3.	$\{T\}\ k := 0;\ x := 1\ \{k = 0 \wedge x = 1\}$	*sequence 1, 2*
4.	$k = 0 \wedge x = 1 \rightarrow k \geq 0 \wedge x = 2\text{^}k$	*predicate logic*
5.	$\{T\}\ k := 0;\ x := 1\ \{k \geq 0 \wedge x = 2\text{^}k\}$	*postcondition weakening 3, 4*

* The weakened or strengthened triple might look nicer than the other. Also, if one of S_1 or S_2 is more painful to write, both proofs involve writing one of S_2 and S_2 once and the other twice.

Example 2

- This is like Example 1 but uses weakest preconditions instead of strongest postconditions.
- The full proof outline is $\{T\}\ \{0 \ge 0 \wedge 1 = 2\hat{}0\}\ k := 0;\ \{k \ge 0 \wedge 1 = 2\hat{}k\}\ x := 1\ \{k \ge 0 \wedge x = 2\hat{}k\}$.

1.	$\{k \ge 0 \wedge 1 = 2\hat{}k\}\ x := 1\ \{k \ge 0 \wedge x = 2\hat{}k\}$	*assignment (backward)*
2.	$\{0 \ge 0 \wedge 1 = 2\hat{}0\}\ k := 0\ \{k \ge 0 \wedge 1 = 2\hat{}k\}$	*assignment (backward)*
3.	$\{0 \ge 0 \wedge 1 = 2\hat{}0\}\ k := 0;\ x := 1\ \{k \ge 0 \wedge x = 2\hat{}k\}$	*sequence 2, 1*
4.	$T \rightarrow 0 \ge 0 \wedge 1 = 2\hat{}0$	*predicate logic*
5.	$\{T\}\ k := 0;\ x := 1\ \{k \ge 0 \wedge x = 2\hat{}k\}$	*precondition strengthening 4, 3*

Example 3

- Here's a full proof outline for the summation loop; note how the structure of the outline follows the partial correctness proof, which is shown below.

$\{n \ge 0\}\ k := 0;\ \{n \ge 0 \wedge k = 0\}\ s := 0;\ \{n \ge 0 \wedge k = 0 \wedge s = 0\}$
$\{\textbf{inv}\ p_1 \equiv 0 \le k \le n \wedge s = sum(0, k)\}$
while $k < n$ ***do***
  $\{p_1 \wedge k < n\}\ \{p_1[k+1/k][s+k+1/s]\}$
  $s := s+k+1;\ \{p_1[k+1/k]\}$
  $k := k+1\ \{p_1\}$
od
$\{p_1 \wedge k \ge n\}$
$\{s = sum(0, n)\}$

- A full proof is below

1.	$\{n \ge 0\}\ k := 0\ \{n \ge 0 \wedge k = 0\}$	*assignment (forward)*
2.	$\{n \ge 0 \wedge k = 0\}\ s := 0\ \{n \ge 0 \wedge k = 0 \wedge s = 0\}$	*assignment (forward)*
3.	$\{n \ge 0\}\ k := 0;\ s := 0\ \{n \ge 0 \wedge k = 0 \wedge s = 0\}$	*sequence 1, 2*
4.	$n \ge 0 \wedge k = 0 \wedge s = 0 \rightarrow p_1$	*predicate logic*
5.	$\{n \ge 0\}\ k := 0;\ s := 0\ \{p_1\}$	*postcondition weakening 3, 4*
6.	$\{p_1[k+1/k]\}\ k := k+1\ \{p_1\}$	*assignment (backward)*
7.	$\{p_1[k+1/k][s+k+1/s]\}\ s := s+k+1\ \{p_1[k+1/k]$	*assignment (backward)*
8.	$\{p_1[k+1/k][s+k+1/s]\}\ s := s+k+1;\ k := k+1\ \{p_1\}$	*sequence 7, 6*
9.	$p_1 \wedge k < n \rightarrow p_1[k+1/k][s+k+1/s]$	*predicate logic*
10.	$\{p_1 \wedge k < n\}\ s := s+k+1;\ k := k+1\ \{p_1\}$	*precondition strength. 9, 8*
11.	$\{\textbf{inv}\ p_1\}\ W\ \{p_1 \wedge k \ge n\}$	*while loop 10*
	where $W \equiv$ ***while*** $k < n$ ***do*** $s := s+k+1;\ k := k+1$ ***od***	
12.	$\{n \ge 0\}\ k := 0;\ s := 0;\ \{\textbf{inv}\ p_1\}\ W\ \{p_1 \wedge k \ge n\}$	*sequence 5,* ***11***
13.	$p_1 \wedge k \ge n \rightarrow s = sum(0, n)$	*predicate logic*
14.	$\{n \ge 0\}\ k := 0;\ s := 0;\ \{\textbf{inv}\ p_1\}\ W\ \{s = sum(0, n)\}$	*postcondition weak. 12, 13*

