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 the invariant, initialization, test, body, and postcondition of a loop are interrelated.
- Know how to write and check a formal proof of partial correctness.

C. Alternative Invariants Yield Different Programs and Proofs

- The invariant, test, initialization code, and body of a loop are all interconnected: Changing one can change them all. For example, we used \( s = \text{sum}(0, k) \) in our earlier summation loop invariant, so we had the loop terminate with \( k = n \).

Example 1: (Original summation loop)

- Using \( s = \text{sum}(0, k) \)
  
  \[
  \left\{ \begin{array}{l}
  n \geq 0 \\
  k := 0; \ s := 0;
  \{ \text{inv } p_1 \equiv 0 \leq k \leq n \land s = \text{sum}(0, k) \}
  \end{array} \right.
  \]
  
  \[
  \text{while } k < n \text{ do}
  \begin{align*}
  &s := s+k+1; \ k := k+1 \\
  \text{od}
  \end{align*}
  \]
  
  \[
  \{ s = \text{sum}(0, n) \}
  \]

- If instead we use \( s = \text{sum}(0, k+1) \) or \( s = \text{sum}(0, k-1) \) in our invariant, we must terminate with \( k+1 = n \) or \( k-1 = n \) respectively, and we change what we should increment \( s \) by.

Example 2: Using \( s = \text{sum}(0, k+1) \)

- Using \( s = \text{sum}(0, k+1) \) gives us
  
  \[
  \left\{ \begin{array}{l}
  n > 0 \\
  k := 0; \ s := 1;
  \{ \text{inv } p_2 \equiv 0 \leq k < n \land s = \text{sum}(0, k+1) \}
  \end{array} \right.
  \]
  
  \[
  \text{while } k < n-1 \text{ do}
  \]

**Proofs and Proof Outlines for Partial Correctness**

*Part 1: Formal Proofs and Full Proof Outlines*

*CS 536: Science of Programming, Fall 2020*
\[ s := s + k + 2; \ k := k + 1 \]

\textit{od}

\{ s = \text{sum}(0, n) \}

\textbf{Example 3: Using} \( s = \text{sum}(0, k-1) \)

- Using \( s = \text{sum}(0, k-1) \) gives us

\[
\begin{align*}
\{ n \geq 0 \} \\
\ k := 0; \ s := 0; \\
\{ \text{inv} \ p_2 = 1 \leq k \leq n+1 \land s = \text{sum}(0, k-1) \} \\
\textbf{while} \ k \leq n \ \textbf{do} \\
\quad s := s+k; \ k := k+1 \\
\textbf{od} \\
\{ s = \text{sum}(0, n) \}
\end{align*}
\]

\textbf{D. 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. \( \text{Length of } AB = \text{length of } XY \) \hspace{1cm} \text{Assumption}
2. \( \text{Angle } ABC = \text{Angle } XYZ \) \hspace{1cm} \text{Assumption}
3. \( \text{Length of } BC = \text{length of } YZ \) \hspace{1cm} \text{Assumption}
4. \( \text{Triangles } ABC, XYZ \text{ are congruent} \) \hspace{1cm} \text{Side-Angle-Side, lines 1, 2, 3}

\textbf{E. 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:

\textbf{Example 1 (repeated):}

\[
\begin{align*}
\{ n \geq 0 \} \\
\ k := 0; \ s := 0; \\
\{ \text{inv} \ p_1 = 0 \leq k \leq n \land s = \text{sum}(0, k) \} \\
\textbf{while} \ k < n \ \textbf{do} \\
\quad s := s+k+1; \ k := k+1 \\
\textbf{od} \\
\{ s = \text{sum}(0, n) \}
\end{align*}
\]

- Below, let \( S_1 = s := s + k + 1; \ k := k + 1 \) (the loop body) and let \( W = \textbf{while} \ k < n \ \textbf{do} \ S_1 \ \textbf{od} \) (the loop).
A in a proof outline, we add conditions to the inside of the program, not just the ends.

Formal proofs are long and contain repetitive information (we keep copying the same conditions over and over).

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).

F. Full Proof Outlines

Formal proofs are long and contain repetitive information (we keep copying the same conditions over and over).

In a proof outline, we add conditions to the inside of the program, not just the ends.

A full proof outline is a way to write out all the information that you would need to generate a formal proof of partial correctness.
• Each triple in the proof must appear in the outline. Said another way, every statement must be part of a triple (including sequences, conditionals, and loops), and every triple must be provable using the proof rules.

• Precondition strengthening appears as a triple with an extra precondition; postcondition weakening appears as a triple with an extra postcondition.

• If two conditions sit next to each other, as in \( \{ p_1 \} \{ p_2 \} \), it stands for a predicate calculus obligation of \( p_1 \rightarrow p_2 \).

• A proof outline does not stand for a unique proof. Aside from permuting line orderings, the timing of precondition strengthening and postcondition weakening may not be unique.

**Example 1**

• Here we form the sequence \( k := 0; x := 1 \) and then weaken its postcondition.

  1. \( \{ T \} \ k := 0 \ \{ k = 0 \} \) Assignment

  2. \( \{ k = 0 \} \ x := 1 \ \{ k = 0 \land x = 1 \} \) Assignment

  3. \( \{ T \} \ k := 0; x := 1 \ \{ k = 0 \land x = 1 \} \) Sequence 1, 2

  4. \( k = 0 \land x = 1 \rightarrow k \geq 0 \land x = 2^k \) Predicate logic

  5. \( \{ T \} \ k := 0; x := 1 \ \{ k \geq 0 \land x = 2^k \} \) Postcond. weak. 3, 4

• The full proof outline for this is \( \{ T \} \ k := 0; \{ k = 0 \} \ x := 1 \ \{ k = 0 \land x = 1 \} \ \{ k \geq 0 \land x = 2^k \} \)

**Example 2**

• This is like Example 1 but uses precondition strengthening instead of postcondition weakening.

  1. \( \{ k \geq 0 \land 1 = 2^k \} \ x := 1 \ \{ k \geq 0 \land x = 2^k \} \) Assignment

  2. \( \{ 0 \geq 0 \land 1 = 2^0 \} \ k := 0 \ \{ k \geq 0 \land 1 = 2^k \} \) Assignment

  3. \( \{ 0 \geq 0 \land 1 = 2^0 \} \ k := 0; x := 1 \ \{ k \geq 0 \land x = 2^k \} \) Sequence 2, 1

  4. \( T \rightarrow 0 \geq 0 \land 1 = 2^0 \) Predicate logic

  5. \( \{ T \} \ k := 0; x := 1 \ \{ k \geq 0 \land x = 2^k \} \) Precond. Str. 4, 3

• The full proof outline is \( \{ T \} \ \{ 0 \geq 0 \land 1 = 2^0 \} \ k := 0; \{ k \geq 0 \land 1 = 2^k \} \ x := 1 \ \{ k \geq 0 \land x = 2^k \} \)

**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.

\[
\begin{align*}
\{ n \geq 0 \} & \ k := 0; \{ n \geq 0 \land k = 0 \} \ s := 0; \{ n \geq 0 \land k = 0 \land s = 0 \} \\
\text{inv } p_1 & = 0 \leq k \leq n \land s = \text{sum}(0, k) \\
\text{while } k < n \text{ do} & \\
\{ p_1 \land k < n \} & \{ p_1[k+1/k][s+k+1/s] \} \\
\text{s := s+k+1; } & \{ p_1[k+1/k] \} \\
k := & k+1 \ \{ p_1 \} \\
\od & \\
\{ p_1 \land k \geq n \} & \\
\{ s = \text{sum}(0, n) \} \\
\end{align*}
\]

• The triples and predicate obligations from this outline are below. They have been listed in the same order as above, in the formal proof of the summation program.
1. \( \{ n \geq 0 \} \ k := 0 \ \{ n \geq 0 \land k = 0 \} \)
2. \( \{ n \geq 0 \land k = 0 \} \ s := 0 \ \{ n \geq 0 \land k = 0 \land s = 0 \} \)
3. \( \{ n \geq 0 \} \ k := 0; \ s := 0 \ \{ n \geq 0 \land k = 0 \land s = 0 \} \)
4. \( n \geq 0 \land k = 0 \land s = 0 \rightarrow p_1 \)
5. \( \{ n \geq 0 \} \ k := 0; \ s := 0 \ \{ p_1 \} \)
6. \( \{ p_1[k+1/k] \} \ k := k+1 \ \{ p_1 \} \)
7. \( \{ p_1[k+1/k][s+k+1/s] \} \ s := s+k+1 \ \{ p_1[k+1/k] \} \)
8. \( \{ p_1[k+1/k][s+k+1/s] \} \ s := s+k+1; \ k := k+1 \ \{ p_1 \} \)
9. \( p_1 \land k < n \rightarrow p_1[k+1/k][s+k+1/s] \)
10. \( \{ p_1 \land k < n \} \ s := s+k+1; \ k := k+1 \ \{ p_1 \} \)
11. \( \{ \text{inv } p_1 \} \ W \ \{ p_1 \land k \geq n \} \) where \( W = \text{while } k < n \ \text{do } s := s+k+1; \ k := k+1 \ \text{od} \)
12. \( \{ n \geq 0 \} \ k := 0; \ s := 0; \ \{ \text{inv } p_1 \} \ W \ \{ p_1 \land k \geq n \} \)
13. \( p_1 \land k \geq n \rightarrow s = \text{sum}(0, n) \)
14. \( \{ n \geq 0 \} \ k := 0; \ s := 0; \ \{ \text{inv } p_1 \} \ W \ \{ s = \text{sum}(0, n) \} \)