Lecture 14: Hoare Logic and Verification Condition Generation

17-355/17-655/17-819: Program Analysis

Rohan Padhye and Jonathan Aldrich

March 23, 2021

* Course materials developed with Claire Le Goues

Review from Last Week

- Axiomatic Semantics: reasoning about code using logical assertions
- Hoare Logic: a logic for proving programs correct
  - Often using Hoare Triples: $\{\ P\ \}\ S\ \{\ Q\ \}$
- Automation using Weakest Preconditions
  - From a postcondition and a statement, compute the weakest precondition that makes a valid Hoare Triple

Review: Weakest Precondition Rules

- $wp(x := E, P) = [E/x]\ P$
- $wp(S;T, Q) = wp(S, wp(T, Q))$
- $wp(\text{if } B \text{ then } S \text{ else } T, Q) = B \Rightarrow wp(S,Q)\ \&\&\ \neg B \Rightarrow wp(T,Q)$

Proving loops correct

- *Partial correctness*
  - The loop may not terminate, but if it does, the postcondition will hold
- {P} while B do S {Q}
  - Find an invariant Inv such that:
    - $P \Rightarrow Inv$
      - The invariant is initially true
    - { Inv && B } S {Inv}
      - Each execution of the loop preserves the invariant
    - $(Inv \ \&\& \ \neg B) \Rightarrow Q$
      - The invariant and the loop exit condition imply the postcondition

Loop Example

- Prove array sum correct

$\{ N \geq 0 \}$

j := 0;

s := 0;

How can we find a loop invariant?

while (j < N) do

    j := j + 1;

    s := s + a[j];

Replace N with j

Add information on range of j

Result: $0 \leq j \leq N$ && $s = (\Sigma i \mid 0 \leq i < j \bullet a[i])$

end

$\{ s = (\Sigma i \mid 0 \leq i < N \bullet a[i]) \}$

Loop Example

- Prove array sum correct

$\{ N \geq 0 \}$

j := 0;

s := 0;

$\{ 0 \leq j \leq N \ \&\&\ s = (\Sigma i \mid 0 \leq i < j \bullet a[i]) \}$

while (j < N) do

  $\{0 \leq j \leq N \ \&\&\ s = (\Sigma i \mid 0 \leq i < j \bullet a[i]) \ \&\&\ j < N\}$

  j := j + 1;

  s := s + a[j];

  $\{0 \leq j \leq N \ \&\&\ s = (\Sigma i \mid 0 \leq i < j \bullet a[i]) \}$

end

$\{ s = (\Sigma i \mid 0 \leq i < N \bullet a[i]) \}$

Loop Example

- Prove array sum correct

{ $N \geq 0$ }
j := 0;
s := 0;
{ $0 \leq j \leq N$ && $s = (\Sigma i \mid 0 \leq i < j \bullet a[i])$ }

Proof obligation #1

while (j < N) do
    {$0 \leq j \leq N$ && $s = (\Sigma i \mid 0 \leq i < j \bullet a[i])$ && $j < N$}
    j := j + 1;
    s := s + a[j];
    {$0 \leq j \leq N$ && $s = (\Sigma i \mid 0 \leq i < j \bullet a[i])$ }

Proof obligation #2

end

Proof obligation #3

{ $s = (\Sigma i \mid 0 \leq i < N \bullet a[i])$ }

Last time, we showed how to use weakest preconditions to verify this proof obligation

Proof Obligations

- Invariant is maintained

  {0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) && j < N}

  ```
  j := j + 1;

  s := s + a[j];
  ```

  {0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) }

Proof Obligations

- Invariant is maintained

  {0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) && j < N}

  j := j + 1;

  {0 ≤ j ≤ N && **s+a[j]** = (Σi | 0≤i<j • a[i]) } *// by assignment rule*

  s := s + a[j];

  {0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) }

Proof Obligations

- Invariant is maintained

  {0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) && j < N}

  {0 ≤ **j +1** ≤ N && s+a[**j+1**] = (Σi | 0≤i<**j+1** • a[i]) } *// by assignment rule*

  j := j + 1;

  {0 ≤ j ≤ N && **s+a[j]** = (Σi | 0≤i<j • a[i]) } *// by assignment rule*

  s := s + a[j];

  {0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) }

Proof Obligations

- Invariant is maintained

  $\{0 \le j \le N \;\&\&\; s = (\Sigma i \mid 0 \le i < j \bullet a[i]) \;\&\&\; j < N\}$

  $\{0 \le j+1 \le N \;\&\&\; s+a[j+1] = (\Sigma i \mid 0 \le i < j+1 \bullet a[i])\}$ *// by assignment rule*

  j := j + 1;

  $\{0 \le j \le N \;\&\&\; s+a[j] = (\Sigma i \mid 0 \le i < j \bullet a[i])\}$ *// by assignment rule*

  s := s + a[j];

  $\{0 \le j \le N \;\&\&\; s = (\Sigma i \mid 0 \le i < j \bullet a[i])\}$

- Need to show that:

  $(0 \le j \le N \;\&\&\; s = (\Sigma i \mid 0 \le i < j \bullet a[i]) \;\&\&\; j < N)$

  $\Rightarrow (0 \le j+1 \le N \;\&\&\; s+a[j+1] = (\Sigma i \mid 0 \le i < j+1 \bullet a[i]))$

Proof Obligations

- Invariant is maintained

  $\{0 \leq j \leq N$ && $s = (\Sigma i \mid 0 \leq i < j \bullet a[i])$ && $j < N\}$

  $\{0 \leq \mathbf{j+1} \leq N$ && $s+a[\mathbf{j+1}] = (\Sigma i \mid 0 \leq i < \mathbf{j+1} \bullet a[i])\}$ *// by assignment rule*

  j := j + 1;

  $\{0 \leq j \leq N$ && $\mathbf{s+a[j]} = (\Sigma i \mid 0 \leq i < j \bullet a[i])\}$ *// by assignment rule*

  s := s + a[j];

  $\{0 \leq j \leq N$ && $s = (\Sigma i \mid 0 \leq i < j \bullet a[i])\}$

- Need to show that:

  $(0 \leq j \leq N$ && $s = (\Sigma i \mid 0 \leq i < j \bullet a[i])$ && $j < N)$

  $\quad\Rightarrow (0 \leq j+1 \leq N$ && $s+a[j+1] = (\Sigma i \mid 0 \leq i < j+1 \bullet a[i]))$

= $(0 \leq j < N$ && $s = (\Sigma i \mid 0 \leq i < j \bullet a[i]))$

  $\quad\Rightarrow (\mathbf{-1 \leq j < N}$ && $s+a[j+1] = (\Sigma i \mid 0 \leq i < j+1 \bullet a[i]))$ *// simplify bounds of j*

Proof Obligations

- Invariant is maintained

  $\{0 \le j \le N$ && $s = (\Sigma i \mid 0 \le i < j \bullet a[i])$ && $j < N\}$

  $\{0 \le \mathbf{j+1} \le N$ && $s+a[\mathbf{j+1}] = (\Sigma i \mid 0 \le i < \mathbf{j+1} \bullet a[i])\}$ *// by assignment rule*

  j := j + 1;

  $\{0 \le j \le N$ && $\mathbf{s+a[j]} = (\Sigma i \mid 0 \le i < j \bullet a[i])\}$ *// by assignment rule*

  s := s + a[j];

  $\{0 \le j \le N$ && $s = (\Sigma i \mid 0 \le i < j \bullet a[i])\}$

- Need to show that:

  $(0 \le j \le N$ && $s = (\Sigma i \mid 0 \le i < j \bullet a[i])$ && $j < N)$

  $\Rightarrow (0 \le j+1 \le N$ && $s+a[j+1] = (\Sigma i \mid 0 \le i < j+1 \bullet a[i]))$

= $(0 \le j < N$ && $s = (\Sigma i \mid 0 \le i < j \bullet a[i]))$

  $\Rightarrow (\mathbf{-1 \le j < N}$ && $s+a[j+1] = (\Sigma i \mid 0 \le i < j+1 \bullet a[i]))$ *// simplify bounds of j*

= $(0 \le j < N$ && $s = (\Sigma i \mid 0 \le i < j \bullet a[i]))$

  $\Rightarrow (-1 \le j < N$ && $s+a[j+1] = (\Sigma i \mid 0 \le i < j \bullet a[i]) + \mathbf{a[j]})$ *// separate last element*

Proof Obligations

- Invariant is maintained

  $\{0 \le j \le N \ \&\& \ s = (\Sigma i \mid 0 \le i < j \bullet a[i]) \ \&\& \ j < N\}$

  $\{0 \le \mathbf{j+1} \le N \ \&\& \ s+a[\mathbf{j+1}] = (\Sigma i \mid 0 \le i < \mathbf{j+1} \bullet a[i])\ \}$ *// by assignment rule*

  $j := j + 1;$

  $\{0 \le j \le N \ \&\& \ \mathbf{s+a[j]} = (\Sigma i \mid 0 \le i < j \bullet a[i])\ \}$ *// by assignment rule*

  $s := s + a[j];$

  $\{0 \le j \le N \ \&\& \ s = (\Sigma i \mid 0 \le i < j \bullet a[i])\ \}$

- Need to show that:

  $(0 \le j \le N \ \&\& \ s = (\Sigma i \mid 0 \le i < j \bullet a[i]) \ \&\& \ j < N)$

  $\quad \Rightarrow (0 \le j+1 \le N \ \&\& \ s+a[j+1] = (\Sigma i \mid 0 \le i < j+1 \bullet a[i]))$

$=$ $(0 \le j < N \ \&\& \ s = (\Sigma i \mid 0 \le i < j \bullet a[i]))$

  $\quad \Rightarrow (\mathbf{-1 \le j < N} \ \&\& \ s+a[j+1] = (\Sigma i \mid 0 \le i < j+1 \bullet a[i]))$ *// simplify bounds of j*

$=$ $(0 \le j < N \ \&\& \ s = (\Sigma i \mid 0 \le i < j \bullet a[i]))$

  $\quad \Rightarrow (-1 \le j < N \ \&\& \ s+\boxed{a[j+1]} = (\Sigma i \mid 0 \le i < j \bullet a[i]) + \boxed{a[j]})$ *// separate last element*

// we have a problem – we need a[j+1] and a[j] to cancel out

Where's the error?

- Prove array sum correct

```
{ N ≥ 0 }
j := 0;
s := 0;

while (j < N) do

    j := j + 1;
    s := s + a[j];

end
{ s = (Σi | 0≤i<N • a[i]) }
```

Where's the error?

- Prove array sum correct

$\{ N \geq 0 \}$

```
j := 0;
s := 0;

while (j < N) do

    j := j + 1;
    s := s + a[j];

end
```

$\{ s = (\Sigma i \mid 0 \leq i < N \bullet a[i]) \}$

Need to add element ***before*** incrementing j

Corrected Code

- Prove array sum correct

$\{ N \geq 0 \}$

```
j := 0;
s := 0;

while (j < N) do

    s := s + a[j];
    j := j + 1;

end
```

$\{ s = (\Sigma i \mid 0 \leq i < N \bullet a[i]) \}$

Proof Obligations

- Invariant is maintained

  {0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) && j < N}

  s := s + a[j];

  j := j + 1;

  {0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) }

Proof Obligations

- Invariant is maintained

  {0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) && j < N}

  s := s + a[j];

  {0 ≤ j **+1** ≤ N && s = (Σi | 0≤i<j**+1** • a[i]) } *// by assignment rule*

  j := j + 1;

  {0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) }

Proof Obligations

- Invariant is maintained

  {$0 \le j \le N$ && $s = (\Sigma i \mid 0 \le i < j \bullet a[i])$ && $j < N$}

  {$0 \le j+1 \le N$ && $\mathbf{s+a[j]} = (\Sigma i \mid 0 \le i < j+1 \bullet a[i])$ } *// by assignment rule*

  s := s + a[j];

  {$0 \le \mathbf{j+1} \le N$ && $s = (\Sigma i \mid 0 \le i < \mathbf{j+1} \bullet a[i])$ } *// by assignment rule*

  j := j + 1;

  {$0 \le j \le N$ && $s = (\Sigma i \mid 0 \le i < j \bullet a[i])$ }

Proof Obligations

- Invariant is maintained

  {0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) && j < N}

  {0 ≤ j +1 ≤ N && **s+a[j]** = (Σi | 0≤i<j+1 • a[i]) } *// by assignment rule*

  s := s + a[j];

  {0 ≤ **j +1** ≤ N && s = (Σi | 0≤**i<j+1** • a[i]) } *// by assignment rule*

  j := j + 1;

  {0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) }

- Need to show that:

  (0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) && j < N) ⇒ (0 ≤ j +1 ≤ N && s+a[j] = (Σi | 0≤i<j+1 • a[i]))

Proof Obligations

- Invant is maintained

  $\{0 \le j \le N \ \&\& \ s = (\Sigma i \mid 0 \le i < j \bullet a[i]) \ \&\& \ j < N\}$

  $\{0 \le j+1 \le N \ \&\& \ \mathbf{s+a[j]} = (\Sigma i \mid 0 \le i < j+1 \bullet a[i])\}$ *// by assignment rule*

  s := s + a[j];

  $\{0 \le \mathbf{j+1} \le N \ \&\& \ s = (\Sigma i \mid 0 \le i < \mathbf{j+1} \bullet a[i])\}$ *// by assignment rule*

  j := j + 1;

  $\{0 \le j \le N \ \&\& \ s = (\Sigma i \mid 0 \le i < j \bullet a[i])\}$

- Need to show that:

  $(0 \le j \le N \ \&\& \ s = (\Sigma i \mid 0 \le i < j \bullet a[i]) \ \&\& \ j < N) \Rightarrow (0 \le j+1 \le N \ \&\& \ s+a[j] = (\Sigma i \mid 0 \le i < j+1 \bullet a[i]))$

$=$ $(0 \le j < N \ \&\& \ s = (\Sigma i \mid 0 \le i < j \bullet a[i])) \Rightarrow (\mathbf{-1 \le j < N} \ \&\& \ s+a[j] = (\Sigma i \mid 0 \le i < j+1 \bullet a[i]))$ *// simplify bounds of j*

Proof Obligations

- Invariant is maintained

  $\{0 \le j \le N$ && $s = (\Sigma i \mid 0 \le i < j \bullet a[i])$ && $j < N\}$

  $\{0 \le j+1 \le N$ && $\mathbf{s+a[j]} = (\Sigma i \mid 0 \le i < j+1 \bullet a[i])\}$ *// by assignment rule*

  s := s + a[j];

  $\{0 \le \mathbf{j+1} \le N$ && $s = (\Sigma i \mid 0 \le i < \mathbf{j+1} \bullet a[i])\}$ *// by assignment rule*

  j := j + 1;

  $\{0 \le j \le N$ && $s = (\Sigma i \mid 0 \le i < j \bullet a[i])\}$

- Need to show that:

  $(0 \le j \le N$ && $s = (\Sigma i \mid 0 \le i < j \bullet a[i])$ && $j < N) \Rightarrow (0 \le j+1 \le N$ && $s+a[j] = (\Sigma i \mid 0 \le i < j+1 \bullet a[i]))$

= $(0 \le j < N$ && $s = (\Sigma i \mid 0 \le i < j \bullet a[i])) \Rightarrow (\mathbf{-1 \le j < N}$ && $s+a[j] = (\Sigma i \mid 0 \le i < j+1 \bullet a[i]))$ *// simplify bounds of j*

= $(0 \le j < N$ && $s = (\Sigma i \mid 0 \le i < j \bullet a[i])) \Rightarrow (-1 \le j < N$ && $s+a[j] = (\Sigma i \mid 0 \le i < j \bullet a[i]) \mathbf{+ a[j]})$ *// separate last part of sum*

Proof Obligations

- Invariant is maintained

  $\{0 \le j \le N \;\&\&\; s = (\Sigma i \mid 0 \le i < j \bullet a[i]) \;\&\&\; j < N\}$

  $\{0 \le j + 1 \le N \;\&\&\; \mathbf{s+a[j]} = (\Sigma i \mid 0 \le i < j+1 \bullet a[i])\}$ *// by assignment rule*

  s := s + a[j];

  $\{0 \le \mathbf{j+1} \le N \;\&\&\; s = (\Sigma i \mid 0 \le i < \mathbf{j+1} \bullet a[i])\}$ *// by assignment rule*

  j := j + 1;

  $\{0 \le j \le N \;\&\&\; s = (\Sigma i \mid 0 \le i < j \bullet a[i])\}$

- Need to show that:

  $(0 \le j \le N \;\&\&\; s = (\Sigma i \mid 0 \le i < j \bullet a[i]) \;\&\&\; j < N) \Rightarrow (0 \le j + 1 \le N \;\&\&\; s+a[j] = (\Sigma i \mid 0 \le i < j+1 \bullet a[i]))$

$=$ $(0 \le j < N \;\&\&\; s = (\Sigma i \mid 0 \le i < j \bullet a[i])) \Rightarrow (\mathbf{-1 \le j < N} \;\&\&\; s+a[j] = (\Sigma i \mid 0 \le i < j+1 \bullet a[i]))$ *// simplify bounds of j*

$=$ $(0 \le j < N \;\&\&\; s = (\Sigma i \mid 0 \le i < j \bullet a[i])) \Rightarrow (-1 \le j < N \;\&\&\; s+a[j] = (\Sigma i \mid 0 \le i < j \bullet a[i]) + \mathbf{a[j]})$ *// separate last part of sum*

$=$ $(0 \le j < N \;\&\&\; s = (\Sigma i \mid 0 \le i < j \bullet a[i])) \Rightarrow (-1 \le j < N \;\&\&\; s = (\Sigma i \mid 0 \le i < j \bullet a[i]))$ *// subtract a[j] from both sides*

Proof Obligations

- Invariant is maintained

  $\{0 \le j \le N \ \&\&\ s = (\Sigma i \mid 0 \le i < j \bullet a[i]) \ \&\&\ j < N\}$

  $\{0 \le j+1 \le N \ \&\&\ \mathbf{s+a[j]} = (\Sigma i \mid 0 \le i < j+1 \bullet a[i])\}$ *// by assignment rule*

  s := s + a[j];

  $\{0 \le \mathbf{j+1} \le N \ \&\&\ s = (\Sigma i \mid 0 \le i < \mathbf{j+1} \bullet a[i])\}$ *// by assignment rule*

  j := j + 1;

  $\{0 \le j \le N \ \&\&\ s = (\Sigma i \mid 0 \le i < j \bullet a[i])\}$

- Need to show that:

  $(0 \le j \le N \ \&\&\ s = (\Sigma i \mid 0 \le i < j \bullet a[i]) \ \&\&\ j < N) \Rightarrow (0 \le j+1 \le N \ \&\&\ s+a[j] = (\Sigma i \mid 0 \le i < j+1 \bullet a[i]))$

$=$ $(0 \le j < N \ \&\&\ s = (\Sigma i \mid 0 \le i < j \bullet a[i])) \Rightarrow (\mathbf{-1 \le j < N} \ \&\&\ s+a[j] = (\Sigma i \mid 0 \le i < j+1 \bullet a[i]))$ *// simplify bounds of j*

$=$ $(0 \le j < N \ \&\&\ s = (\Sigma i \mid 0 \le i < j \bullet a[i])) \Rightarrow (-1 \le j < N \ \&\&\ s+a[j] = (\Sigma i \mid 0 \le i < j \bullet a[i]) \mathbf{+ a[j]})$ *// separate last part of sum*

$=$ $(0 \le j < N \ \&\&\ s = (\Sigma i \mid 0 \le i < j \bullet a[i])) \Rightarrow (-1 \le j < N \ \&\&\ s = (\Sigma i \mid 0 \le i < j \bullet a[i]))$ *// subtract a[j] from both sides*

$=$ true *//* $0 \le j \Rightarrow -1 \le j$

Proof Obligations

- Invariant and exit condition implies postcondition

$0 \le j \le N$ && $s = (\Sigma i \mid 0 \le i < j \bullet a[i])$ && $j \ge N$

$\Longrightarrow s = (\Sigma i \mid 0 \le i < N \bullet a[i])$

Proof Obligations

- Invariant and exit condition implies postcondition

$0 \le j \le N \;\&\&\; s = (\Sigma i \mid 0 \le i < j \bullet a[i]) \;\&\&\; j \ge N \Rightarrow s = (\Sigma i \mid 0 \le i < N \bullet a[i])$

$= \quad 0 \le j \;\&\&\; j = N \;\&\&\; s = (\Sigma i \mid 0 \le i < j \bullet a[i]) \Rightarrow s = (\Sigma i \mid 0 \le i < N \bullet a[i])$

// *because* $(j \le N \;\&\&\; j \ge N) = (j = N)$

Proof Obligations

- Invariant and exit condition implies postcondition

$0 \le j \le N \;\&\&\; s = (\Sigma i \mid 0 \le i < j \bullet a[i]) \;\&\&\; j \ge N \Rightarrow s = (\Sigma i \mid 0 \le i < N \bullet a[i])$

$= \quad 0 \le j \;\&\&\; j = N \;\&\&\; s = (\Sigma i \mid 0 \le i < j \bullet a[i]) \Rightarrow s = (\Sigma i \mid 0 \le i < N \bullet a[i])$

// *because* $(j \le N \;\&\&\; j \ge N) = (j = N)$

$= \quad 0 \le N \;\&\&\; s = (\Sigma i \mid 0 \le i < N \bullet a[i]) \Rightarrow s = (\Sigma i \mid 0 \le i < N \bullet a[i])$

// *by substituting N for j, since j = N*

Proof Obligations

- Invariant and exit condition implies postcondition

$0 \le j \le N \;\&\&\; s = (\Sigma i \mid 0 \le i < j \bullet a[i]) \;\&\&\; j \ge N \Rightarrow s = (\Sigma i \mid 0 \le i < N \bullet a[i])$

$= \;\; 0 \le j \;\&\&\; j = N \;\&\&\; s = (\Sigma i \mid 0 \le i < j \bullet a[i]) \Rightarrow s = (\Sigma i \mid 0 \le i < N \bullet a[i])$

// *because* $(j \le N \;\&\&\; j \ge N) = (j = N)$

$= \;\; 0 \le N \;\&\&\; s = (\Sigma i \mid 0 \le i < N \bullet a[i]) \Rightarrow s = (\Sigma i \mid 0 \le i < N \bullet a[i])$

// *by substituting N for j, since j = N*

= **true** // *because* $P \;\&\&\; Q \Rightarrow Q$

Practice: Writing Proof Obligations

- For the program below and the invariant i <= N, write the proof obligations. The form of your answer should be three mathematical implications.

```
{ N >= 0 }

i := 0;

while (i < N) do

   i := N

{ i = N }
```

- Invariant is initially true:
- Invariant is preserved by the loop body:
- Invariant and exit condition imply postcondition:

Invariant Intuition

- For code without loops, we are simulating execution directly
  - We prove one Hoare Triple for each statement, and each statement is executed once

- For code with loops, we are doing *one* proof of correctness for *multiple* loop iterations
  - Proof must cover all iterations
    - Don't know how many there will be
  - The invariant must be *general* yet *precise*
    - general enough to be true for every execution
    - precise enough to imply the postcondition we need
  - This tension makes inferring loop invariants challenging

Total Correctness for Loops

- {P} while B do S {Q}
- Partial correctness:
  - Find an invariant Inv such that:
    - $P \Rightarrow Inv$
      - The invariant is initially true
    - { Inv && B } S {Inv}
      - Each execution of the loop preserves the invariant
    - $(Inv \ \&\& \ \neg B) \Rightarrow Q$
      - The invariant and the loop exit condition imply the postcondition
- Total correctness
  - Loop will terminate

We haven't proven termination

- Consider the following program:

```
{ true }
i := 0
while (true) do         { true }
   i := i + 1;
{ i == -1 }
```

We haven't proven termination

- Consider the following program:

```
{ true }
i := 0
while (true) do        { true }
   i := i + 1;
{ i == -1 }
```

- This program verifies (as partially correct)
  - Loop invariant trivially true initially and trivially preserved
  - Postcondition check:
    - (not(true) && true) => (i == -1)
    - = (false && true) => (i == -1)
    - = (false) => (i == -1)
    - = true

We haven't proven termination

- Consider the following program:

```
{ true }
i := 0
while (true) do        { true }
   i := i + 1;
{ i == -1 }
```

- This program verifies (as partially correct)
  - Loop invariant trivially true initially and trivially preserved
  - Postcondition check:
    - (not(true) && true) => (i == -1)
    - = (false && true) => (i == -1)
    - = (false) => (i == -1)
    - = true
  - Partial correctness: if the program terminates, then the postcondition will hold
    - Doesn't say anything about the postcondition if the program does not terminate—any postcondition is OK.
    - We need a stronger correctness property

Termination

```
{ N ≥ 0 }
j := 0;
s := 0;

while (j < N) do

    s := s + a[j];
    j := j + 1;

end
{ s = (Σi | 0≤i<N • a[i]) }
```

- How would you prove this program terminates?

Termination

```
{ N ≥ 0 }
j := 0;
s := 0;

while (j < N) do

    s := s + a[j];
    j := j + 1;

end
{ s = (Σi | 0≤i<N • a[i]) }
```

- How would you prove this program terminates?
- Consider the loop
  - What is the maximum number of times it could execute?
  - Use induction to prove this bound is correct

Total Correctness for Loops

- {P} while B do S {Q}
- Partial correctness:
  - Find an invariant Inv such that:
    - $P \Rightarrow Inv$
      - The invariant is initially true
    - { Inv && B } S {Inv}
      - Each execution of the loop preserves the invariant
    - $(Inv \ \&\& \ \neg B) \Rightarrow Q$
      - The invariant and the loop exit condition imply the postcondition
- Termination bound
  - Find a *variant function* v such that:
    - v is an upper bound on the number of loops remaining

Total Correctness for Loops

- {P} while B do S {Q}
- Partial correctness:
  - Find an invariant Inv such that:
    - $P \Rightarrow Inv$
      - The invariant is initially true
    - { Inv && B } S {Inv}
      - Each execution of the loop preserves the invariant
    - $(Inv \ \&\& \ \neg B) \Rightarrow Q$
      - The invariant and the loop exit condition imply the postcondition

- Termination bound
  - Find a *variant function* v such that:
    - v is an upper bound on the number of loops remaining

    - { Inv && B && v=V } S {v < V}
      - The variant function decreases each time the loop body executes

Total Correctness for Loops

- {P} while B do S {Q}
- Partial correctness:
  - Find an invariant Inv such that:
    - $P \Rightarrow Inv$
      - The invariant is initially true
    - { Inv && B } S {Inv}
      - Each execution of the loop preserves the invariant
    - $(Inv \;\&\&\; \neg B) \Rightarrow Q$
      - The invariant and the loop exit condition imply the postcondition
- Termination bound
  - Find a *variant function* v such that:
    - v is an upper bound on the number of loops remaining
    - { Inv && B && v=V } S {v < V}
      - The variant function decreases each time the loop body executes
    - $(Inv \;\&\&\; v \leq 0) \Rightarrow \neg B$
      - If we the variant function reaches zero, we must exit the loop

Total Correctness Example

while (j < N) do

$\{0 \le j \le N \;\&\&\; s = (\Sigma i \mid 0 \le i < j \bullet a[i]) \;\&\&\; j < N\}$

s := s + a[j];

j := j + 1;

$\{0 \le j \le N \;\&\&\; s = (\Sigma i \mid 0 \le i < j \bullet a[i])\}$

end

- Variant function for this loop?

Total Correctness Example

while (j < N) do

$\{0 \leq j \leq N \;\&\&\; s = (\Sigma i \mid 0 \leq i < j \bullet a[i]) \;\&\&\; j < N\}$

s := s + a[j];

j := j + 1;

$\{0 \leq j \leq N \;\&\&\; s = (\Sigma i \mid 0 \leq i < j \bullet a[i])\}$

end

- Variant function for this loop?
  - N-j

Guessing Variant Functions

- Loops with an index
  - $N \pm i$
  - Applies if you always add or always subtract a constant, and if you exit the loop when the index reaches some constant
  - Use $N-i$ if you are incrementing i, $N+i$ if you are decrementing i
  - Set N such that $N \pm i \leq 0$ at loop exit

- Other loops
  - Find an expression that is an upper bound on the number of iterations left in the loop

Additional Proof Obligations

- Variant function for this loop: N-j
- To show: variant function is decreasing

  $\{0 \le j \le N \ \&\&\ s = (\Sigma i \mid 0 \le i < j \bullet a[i]) \ \&\&\ j < N \ \&\&\ N\text{-}j = V\}$

  s := s + a[j];

  j := j + 1;

  $\{N\text{-}j < V\}$

Additional Proof Obligations

- Variant function for this loop: N-j
- To show: variant function is decreasing

  $\{0 \leq j \leq N \text{ \&\& } s = (\Sigma i \mid 0 \leq i < j \bullet a[i]) \text{ \&\& } j < N \text{ \&\& } N-j = V\}$

  s := s + a[j];

  j := j + 1;

  $\{N-j < V\}$

- To show: exit the loop once variant function reaches 0

  $(0 \leq j \leq N \text{ \&\& } s = (\Sigma i \mid 0 \leq i < j \bullet a[i]) \text{ \&\& } N-j \leq 0)$

  $\Rightarrow j \geq N$

Additional Proof Obligations

- To show: variant function is decreasing
  {0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) && j < N && N-j = V}

  s := s + a[j];

  j := j + 1;
  {N-j < V}

Additional Proof Obligations

- To show: variant function is decreasing

  {0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) && j < N && N-j = V}

  s := s + a[j];

  {N-(**j+1**) < V}   *// by assignment*

  j := j + 1;

  {N-j < V}

Additional Proof Obligations

- To show: variant function is decreasing

  {0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) && j < N && N-j = V}

  {N-(j+1) < V} *// by assignment*

  s := s + a[j];

  {N-(**j+1**) < V} *// by assignment*

  j := j + 1;

  {N-j < V}

Additional Proof Obligations

- To show: variant function is decreasing

  {0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) && j < N && N-j = V}

  {N-(j+1) < V} *// by assignment*

  s := s + a[j];

  {N-(j+1) < V} *// by assignment*

  j := j + 1;

  {N-j < V}

- Need to show:

  (0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) && j < N && N-j = V)

  ⇒ (N-(j+1) < V)

Additional Proof Obligations

- To show: variant function is decreasing

  {$0 \leq j \leq N$ && $s = (\Sigma i \mid 0 \leq i < j \bullet a[i])$ && $j < N$ && $N\text{-}j = V$}

  {$N-(j+1) < V$} *// by assignment*

  s := s + a[j];

  {$N-(j+1) < V$} *// by assignment*

  j := j + 1;

  {$N\text{-}j < V$}

- Need to show:

  ($0 \leq j \leq N$ && $s = (\Sigma i \mid 0 \leq i < j \bullet a[i])$ && $j < N$ && $N\text{-}j = V$)

  $\Rightarrow$ ($N-(j+1) < V$)

Assume $0 \leq j \leq N$ && $s = (\Sigma i \mid 0 \leq i < j \bullet a[i])$ && $j < N$ && $N\text{-}j = V$

Additional Proof Obligations

- To show: variant function is decreasing

  {0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) && j < N && N-j = V}

  {N-(j+1) < V} *// by assignment*

  s := s + a[j];

  {N-(j+1) < V} *// by assignment*

  j := j + 1;

  {N-j < V}

- Need to show:

  (0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) && j < N && N-j = V)

  ⇒ (N-(j+1) < V)

Assume 0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) && j < N && N-j = V

By weakening we have N-j = V

Additional Proof Obligations

- To show: variant function is decreasing

  {0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) && j < N && N-j = V}

  {N-(j+1) < V} *// by assignment*

  s := s + a[j];

  {N-(j+1) < V} *// by assignment*

  j := j + 1;

  {N-j < V}

- Need to show:

  (0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) && j < N && N-j = V)

  ⇒ (N-(j+1) < V)

Assume 0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) && j < N && N-j = V

By weakening we have N-j = V

Therefore N-j-1 < V

Additional Proof Obligations

- To show: variant function is decreasing

  $\{0 \leq j \leq N \;\&\&\; s = (\Sigma i \mid 0 \leq i < j \bullet a[i]) \;\&\&\; j < N \;\&\&\; N\text{-}j = V\}$

  $\{N-(j+1) < V\}$ *// by assignment*

  s := s + a[j];

  $\{N-(j+1) < V\}$ *// by assignment*

  j := j + 1;

  $\{N-j < V\}$

- Need to show:

  $(0 \leq j \leq N \;\&\&\; s = (\Sigma i \mid 0 \leq i < j \bullet a[i]) \;\&\&\; j < N \;\&\&\; N\text{-}j = V)$

  $\quad\quad \Rightarrow (N-(j+1) < V)$

Assume $0 \leq j \leq N \;\&\&\; s = (\Sigma i \mid 0 \leq i < j \bullet a[i]) \;\&\&\; j < N \;\&\&\; N\text{-}j = V$

By weakening we have $N\text{-}j = V$

Therefore $N\text{-}j\text{-}1 < V$

But this is equivalent to $N-(j+1) < V$, so we are done.

Additional Proof Obligations

- To show: exit the loop once variant function reaches 0

  $(0 \leq j \leq N \ \&\& \ s = (\Sigma i \mid 0 \leq i < j \bullet a[i]) \ \&\& \ N-j \leq 0)$

  $\quad \Longrightarrow j \geq N$

Additional Proof Obligations

- To show: exit the loop once variant function reaches 0

  $(0 \le j \le N \ \&\& \ s = (\Sigma i \mid 0 \le i < j \bullet a[i]) \ \&\& \ N\text{-}j \le 0)$

  $\Rightarrow j \ge N$

  $(0 \le j \le N \ \&\& \ s = (\Sigma i \mid 0 \le i < j \bullet a[i]) \ \&\& \ \mathbf{N \le j})$

  $\Rightarrow j \ge N$ *// added j to both sides*

Additional Proof Obligations

- To show: exit the loop once variant function reaches 0

  $(0 \le j \le N \ \&\& \ s = (\Sigma i \mid 0 \le i < j \bullet a[i]) \ \&\& \ N\text{-}j \le 0)$

  $\quad \Rightarrow j \ge N$

  $(0 \le j \le N \ \&\& \ s = (\Sigma i \mid 0 \le i < j \bullet a[i]) \ \&\& \ \mathbf{N \le j})$

  $\quad \Rightarrow j \ge N$ *// added j to both sides*

= **true** *// $(N \le j) = (j \ge N)$, $P \ \&\& \ Q \Rightarrow P$*

Practice: Variant Functions

For each of the following loops, is the given variant function correct? If not, why not?

A) Loop:

```
n := 256;
while (n > 1) do
    n := n / 2
```

Variant Function: $\log_2 n$

B) Loop:

```
n := 100;
while (n > 0) do
    if (random())
        then n := n + 1;
        else n := n – 1;
```

Variant Function: n

C) Loop:

```
n := 0;
while (n < 10) do
    n := n + 1;
```

Variant Function: -n

A little more formalism

Semantics of Hoare Triples

- A partial correctness assertion $\models \{P\}\ S\ \{Q\}$ is defined formally to mean:

$$\forall E.\forall E'.(E \models P \wedge \langle E, S\rangle \Downarrow E') \Rightarrow E' \models Q$$

- How would we define total correctness $[P]\ S\ [Q]$?

Derivation Rules for Logical Formulas

- We can define rules for proving the validity of logical formulas
  - $\vdash \ A$ is read “we can prove A
- Example rule: $$\frac{\vdash A \quad \vdash B}{\vdash A \wedge B}\ and$$

Derivation Rules for Hoare Logic

- Judgment form: $\vdash \{P\}\ S\ \{Q\}$ means "we can prove the Hoare triple $\{P\}\ S\ \{Q\}$"

$$\frac{}{\vdash \{P\}\ \text{skip}\ \{P\}}\ \mathit{skip} \qquad \frac{}{\vdash \{[e/x]P\}\ x{:=}e\ \{P\}}\ \mathit{assign}$$

$$\frac{\vdash \{P\}\ S_1\ \{P'\} \qquad \vdash \{P'\}\ S_2\ \{Q\}}{\vdash \{P\}\ S_1;\ S_2\ \{Q\}}\ \mathit{seq}$$

$$\frac{\vdash \{P \wedge b\} S_1 \{Q\} \qquad \vdash \{P \wedge \neg b\}\ S_2\ \{Q\}}{\vdash \{P\}\ \text{if } b \text{ then } S_1 \text{ else } S_2\ \{Q\}}\ \mathit{if}$$

The Rule of Consequence – Assembling Proofs

- What if you don’t have exactly the right precondition or postcondition?
  - The Rule of Consequence shows how you can adjust what you can prove to what you need

$$\frac{\vdash P' \Rightarrow P \qquad \vdash \{P\}\ S\ \{Q\} \qquad \vdash Q \Rightarrow Q'}{\vdash \{P'\}\ S\ \{Q'\}}\ consq$$

Constructing Derivations with the Rule of Consequence

$$\dfrac{\vdash \text{true} \Rightarrow e = e \qquad \overline{\{e = e\}\ x := e\ \{x = e\}}}{\vdash \{\text{true}\}x := e\{x = e\}}$$

Can we also formalize proof obligations?

- Yes, with *verification condition generation*
  - Bonus: we can get one formula for correctness of the whole program
  - Rather than segmenting into several formulas that we prove individually

$$
\begin{array}{ll}
VCGen(\text{skip}, Q) & = \\
VCGen(S_1; S_2, Q) & = \\
VCGen(\text{if } b \text{ then } S_1 \text{ else } S_2, Q) & = \\
VCGen(x := e, Q) & =
\end{array}
$$

Can we also formalize proof obligations?

- Yes, with *verification condition generation*
  - Bonus: we can get one formula for correctness of the whole program
  - Rather than segmenting into several formulas that we prove individually

$$
\begin{array}{lcl}
VCGen(\text{skip}, Q) & = & Q \\
VCGen(S_1; S_2, Q) & = & VCGen(S_1, VCGen(S_2, Q)) \\
VCGen(\text{if } b \text{ then } S_1 \text{ else } S_2, Q) & = & b \Rightarrow VCGen(S_1, Q) \wedge \neg b \Rightarrow VCGen(S_2, Q) \\
VCGen(x := e, Q) & = & [e/x]Q
\end{array}
$$

  - Loops are special—as usual!

$$VCGen(\text{while}_{inv}\ e \text{ do } S, Q) = Inv \wedge (\forall x_1 ... x_n . Inv \Rightarrow (e \Rightarrow VCGen(S, Inv) \wedge \neg e \Rightarrow Q))$$

Verification Condition Generation - Summary & Future Lectures

- **Verification Conditions** make axiomatic semantics practical.
  - We can solve them automatically with SAT solvers
  - We can compute verification conditions **forward** for use on unstructured code (= assembly language). This is sometimes called **symbolic execution**.
- We can add extra invariants or drop paths (dropping is *unsound*) to help verification condition generation **scale**.
- We can model **exceptions**, **memory** operations and **data structures** using verification condition generation.

Heads up: Course Projects

- Scope: ~3 weeks of effort at end of course
- Some options
  - Implement a non-trivial analysis and evaluate it on some code
  - Empirically evaluate an existing analysis tool
  - Contribute meaningfully to an open source analysis tool
  - Explore an extension to the state of the art in program analysis
  - Lots more you can do
- Students in the Ph.D. version (17-819) must engage in research in some way
  - OK to extend your current research work – can be empirical as well