

Lecture 14–15: Hoare Logic

17-355/17-665/17-819: Program Analysis

Rohan Padhye

October 21 & 23, 2025

* Course materials developed with Jonathan Aldrich and Claire Le Goues

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
- Students in the Masters version (17-665) must engage with non-trivial codebases
 - Either the analysis framework or the target program must be in active use by the developer community
- 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

Logical Reasoning about Code

- So far, we've reasoned about code using operational semantics
 - And built program analyses that abstract those semantics
- ***Axiomatic semantics*** define meaning of a program in terms of assertions
 - Enables logic-based reasoning about code
- Enables ***verification***
 - Prove arbitrary properties about code – not just ones built into a particular analysis
 - Goes back to Turing (1949): “Checking a Large Routine”
 - Hoare developed rules in the 1960s for verifying the WHILE language

Axiomatic Semantics

- An **axiomatic semantics** consists of:
 - A **language for stating assertions** about programs,
 - **Rules** for establishing the truth of assertions
- Some typical kinds of assertions:
 - This program terminates
 - If this program terminates, the variables x and y have the same value throughout the execution of the program
 - The array accesses are within the array bounds
- Assertions are in a logic, e.g. first-order logic
 - Alternatives include temporal logic, linear logic, etc.

Assertion Language

$$\begin{array}{l} P ::= \text{true} \quad | \quad \text{false} \quad | \quad e_1 = e_2 \quad | \quad e_1 \geq e_2 \quad | \quad P_1 \wedge P_2 \\ \quad | \quad P_1 \vee P_2 \quad | \quad P_1 \Rightarrow P_2 \quad | \quad \forall x.P \quad | \quad \exists x.P \end{array}$$

- We'll be a bit sloppy and mix logical and program variables like x
- We'll treat Boolean expressions as a special case of assertions

Hoare Triple

$$\{ P \} S \{ Q \}$$

- P is the precondition
- Q is the postcondition
- S is any statement (in WHILE, at least for our class)
- Semantics: if P holds in some state E and if $\langle S; E \rangle \Downarrow E'$, then Q holds in E'
 - This is *partial correctness*: termination of S is not guaranteed
 - *Total correctness* additionally implies termination, and is written $[P] S [Q]$

Exercise: Exploring Hoare Triples

- What are reasonable pre- or post- conditions for the following incomplete Hoare triples?

1. $\{ \text{true} \} \ x := 5 \quad \{ \quad \}$
2. $\{ \quad \} \ x := x + 3 \quad \{ \ x = y + 3 \}$
3. $\{ \quad \} \ x := x * 2 + 3 \quad \{ \ x > 1 \}$
4. $\{ \ x = a \} \ \text{if } (x < 0) \text{ then } x := -x \quad \{ \quad \}$
5. $\{ \text{false} \} \ x := 3 \quad \{ \quad \}$
6. $\{ \ x < 0 \} \ \text{while } (x \neq 0) \ x := x - 1 \quad \{ \quad \}$

Hoare Triple

$$\{ P \} S \{ Q \}$$

- P is the precondition
 - Q is the postcondition
 - S is any statement (in WHILE, at least for our class)
- Semantics: if P holds in some state E and if $\langle S; E \rangle \Downarrow E'$, then Q holds in E'
 - This is *partial correctness*: termination of S is not guaranteed
 - *Total correctness* additionally implies termination, and is written $[P] S [Q]$

Assertion Semantics

- $E \models P$ means P is true in E
- Rules:
 - $E \models \text{true}$ *always*
 - $E \models a_1 = a_2$ *iff* $\langle E, a_1 \rangle \Downarrow n$ and $\langle E, a_2 \rangle \Downarrow n$
 - $E \models a_1 \geq a_2$ *iff* $\langle E, a_1 \rangle \Downarrow n_1, \langle E, a_2 \rangle \Downarrow n_2$, and $n_1 \geq n_2$
 - $E \models P_1 \wedge P_2$ *iff* $E \models P_1$ and $E \models P_2$
 - ...
 - $E \models \forall x.P$ *iff* $\forall n \in \mathbb{Z}. E[x \mapsto n] \models P$
 - $E \models \exists x.P$ *iff* $\exists n \in \mathbb{Z}. E[x \mapsto n] \models P$

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 $[P] S [Q]$?
- This is a good formal definition—but it doesn't help us prove many assertions because we have to reason about all environments. How can we do better?

Derivation Rules for Logical Formulas

- We can define rules for proving the validity of logical formulas
 - $\vdash P$ is read “we can prove P ”
- Example rule:

$$\frac{\vdash P \quad \vdash Q}{\vdash P \wedge Q} \text{ and}$$

Derivation Rules for Hoare Logic

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

$$\frac{}{\vdash \{P\} \text{skip} \{P\}} \text{skip} \quad \frac{}{\vdash \{[a/x]P\} x:=a \{P\}} \text{assign}$$

$$\frac{\vdash \{P\} S_1 \{P'\} \quad \vdash \{P'\} S_2 \{Q\}}{\vdash \{P\} S_1; S_2 \{Q\}} \text{seq} \quad \frac{\vdash \{P \wedge b\} S_1 \{Q\} \quad \vdash \{P \wedge \neg b\} S_2 \{Q\}}{\vdash \{P\} \text{if } b \text{ then } S_1 \text{ else } S_2 \{Q\}} \text{if}$$

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

- Question: What should be the rule for while b do S ?

Soundness and Completeness

- Sound: **if** $\vdash \{P\}S\{Q\}$ **then** $\models \{P\}S\{Q\}$
- Complete: **if** $\models \{P\}S\{Q\}$ **then** $\vdash \{P\}S\{Q\}$

Strongest Postconditions

- Here are a number of valid Hoare Triples:
 - $\{x = 5\} x := x * 2 \{ \text{true} \}$
 - $\{x = 5\} x := x * 2 \{ x > 0 \}$
 - $\{x = 5\} x := x * 2 \{ x = 10 \mid \mid x = 5 \}$
 - $\{x = 5\} x := x * 2 \{ x = 10 \}$
- Which one is best?

Strongest Postconditions

- Here are a number of valid Hoare Triples:
 - $\{x = 5\} x := x * 2 \{ \text{true} \}$
 - $\{x = 5\} x := x * 2 \{ x > 0 \}$
 - $\{x = 5\} x := x * 2 \{ x = 10 \mid \mid x = 5 \}$
 - $\{x = 5\} x := x * 2 \{ x = 10 \}$
 - All are true, but this one is the most *useful*
 - $x=10$ is the *strongest postcondition*
- If $\{P\} S \{Q\}$ and for all Q' such that $\{P\} S \{Q'\}$, $Q \Rightarrow Q'$, then Q is the strongest postcondition of S with respect to P
 - check: $x = 10 \Rightarrow \text{true}$
 - check: $x = 10 \Rightarrow x > 0$
 - check: $x = 10 \Rightarrow x = 10 \mid \mid x = 5$
 - check: $x = 10 \Rightarrow x = 10$

Weakest Preconditions

- Here are a number of valid Hoare Triples:
 - $\{x = 5 \ \&\& \ y = 10\} \quad z := x / y \quad \{z < 1\}$
 - $\{x < y \ \&\& \ y > 0\} \quad z := x / y \quad \{z < 1\}$
 - $\{y \neq 0 \ \&\& \ x / y < 1\} \quad z := x / y \quad \{z < 1\}$
- Which one is best?

Weakest Preconditions

- Here are a number of valid Hoare Triples:
 - $\{x = 5 \ \&\& \ y = 10\} \quad z := x / y \quad \{z < 1\}$
 - $\{x < y \ \&\& \ y > 0\} \quad z := x / y \quad \{z < 1\}$
 - $\{y \neq 0 \ \&\& \ x / y < 1\} \quad z := x / y \quad \{z < 1\}$
 - All are true, but this one is the most *useful* because it allows us to invoke the program in the most general condition
 - $y \neq 0 \ \&\& \ x / y < 1$ is the *weakest precondition*
- If $\{P\} S \{Q\}$ and for all P' such that $\{P'\} S \{Q\}$, $P' \Rightarrow P$, then P is the weakest precondition $wp(S, Q)$ of S with respect to Q

Hoare Triples and Weakest Preconditions

- Theorem: $\{P\} S \{Q\}$ holds if and only if $P \Rightarrow wp(S, Q)$
 - In other words, a Hoare Triple is still valid if the precondition is stronger than necessary, but not if it is too weak
 - Can use this to prove $\{P\} S \{Q\}$ by computing $wp(S, Q)$ and checking implication.
- Question: Could we state a similar theorem for a strongest postcondition function?
 - e.g. $\{P\} S \{Q\}$ holds if and only if $sp(S, P) \Rightarrow Q$
 - A: Yes, but it's harder to compute (see text for why)

Exercise: More Hoare Triples

Consider the following Hoare triples:

A) $\{ z = y + 1 \} x := z * 2 \{ x = 4 \}$

B) $\{ y = 7 \} x := y + 3 \{ x > 5 \}$

C) $\{ \text{false} \} x := 2 / y \{ \text{true} \}$

D) $\{ y < 16 \} x := y / 2 \{ x < 8 \}$

- Which of the Hoare triples above are valid?
- Considering the valid Hoare triples, for which ones can you write a stronger postcondition? (Leave the precondition unchanged, and ensure the resulting triple is still valid)
- Considering the valid Hoare triples, for which ones can you write a weaker precondition? (Leave the postcondition unchanged, and ensure the resulting triple is still valid)

Hoare Logic Rules

- Assignment
 - $\{ P \} x := 3 \{ x+y > 0 \}$
 - What is the weakest precondition P?

Hoare Logic Rules

- Assignment
 - $\{ P \} x := 3 \{ x+y > 0 \}$
 - What is the weakest precondition P?
 - What is most general value of y such that $3 + y > 0$?
 - $y > -3$

Hoare Logic Rules

- Assignment
 - $\{ P \} x := 3 \{ x+y > 0 \}$
 - What is the weakest precondition P ?
- Assignment rule
 - $wp(x := e, P) = [e/x] P$
 - Resulting triple: $\{ [e/x] P \} x := e \{ P \}$

Hoare Logic Rules

- Assignment
 - $\{ P \} x := 3 \{ x+y > 0 \}$
 - What is the weakest precondition P?
- Assignment rule
 - $wp(x := e, P) = [e/x] P$
 - Resulting triple: $\{ [e/x] P \} x := e \{ P \}$
 - $[3 / x] (x + y > 0)$
 - $= (3) + y > 0$
 - $= y > -3$

Hoare Logic Rules

- Assignment
 - $\{ P \} x := 3*y + z \{ x * y - z > 0 \}$
 - What is the weakest precondition P?

Hoare Logic Rules

- Assignment
 - $\{ P \} x := 3*y + z \{ x * y - z > 0 \}$
 - What is the weakest precondition P?
- Assignment rule
 - $wp(x := e, P) = [e/x] P$

Hoare Logic Rules

- Assignment
 - $\{ P \} x := 3*y + z \{ x * y - z > 0 \}$
 - What is the weakest precondition P?
- Assignment rule
 - $wp(x := e, P) = [e/x] P$
 - $[3*y+z / x] (x * y - z > 0)$

Hoare Logic Rules

- Assignment
 - $\{ P \} x := 3*y + z \{ x * y - z > 0 \}$
 - What is the weakest precondition P?
- Assignment rule
 - $wp(x := e, P) = [e/x] P$
 - $[3*y+z / x] (x * y - z > 0)$
 - $= (3*y+z) * y - z > 0$

Hoare Logic Rules

- Assignment
 - $\{ P \} x := 3*y + z \{ x * y - z > 0 \}$
 - What is the weakest precondition P?
- Assignment rule
 - $wp(x := e, P) = [e/x] P$
 - $[3*y+z / x] (x * y - z > 0)$
 - $= (3*y+z) * y - z > 0$
 - $= 3*y^2 + z*y - z > 0$

Hoare Logic Rules

- Sequence
 - $\{ P \} x := x + 1; y := x + y \{ y > 5 \}$
 - What is the weakest precondition P?

Hoare Logic Rules

- Sequence
 - $\{ P \} x := x + 1; y := x + y \{ y > 5 \}$
 - What is the weakest precondition P?
- Sequence rule
 - $wp(S;T, Q) = wp(S, wp(T, Q))$
 - $wp(x:=x+1; y:=x+y, y>5)$

Hoare Logic Rules

- Sequence
 - $\{ P \} x := x + 1; y := x + y \{ y > 5 \}$
 - What is the weakest precondition P?
- Sequence rule
 - $wp(S;T, Q) = wp(S, wp(T, Q))$
 - $wp(x:=x+1; y:=x+y, y>5)$
 - $= wp(x:=x+1, wp(y:=x+y, y>5))$

Hoare Logic Rules

- Sequence
 - $\{ P \} x := x + 1; y := x + y \{ y > 5 \}$
 - What is the weakest precondition P?
- Sequence rule
 - $wp(S;T, Q) = wp(S, wp(T, Q))$
 - $wp(x:=x+1; y:=x+y, y>5)$
 - $= wp(x:=x+1, wp(y:=x+y, y>5))$
 - $= wp(x:=x+1, x+y>5)$

Hoare Logic Rules

- Sequence
 - $\{ P \} x := x + 1; y := x + y \{ y > 5 \}$
 - What is the weakest precondition P?
- Sequence rule
 - $wp(S;T, Q) = wp(S, wp(T, Q))$
 - $wp(x:=x+1; y:=x+y, y>5)$
 - $= wp(x:=x+1, wp(y:=x+y, y>5))$
 - $= wp(x:=x+1, x+y>5)$
 - $= x+1+y>5$

Hoare Logic Rules

- Sequence
 - $\{ P \} x := x + 1; y := x + y \{ y > 5 \}$
 - What is the weakest precondition P?
- Sequence rule
 - $wp(S;T, Q) = wp(S, wp(T, Q))$
 - $wp(x:=x+1; y:=x+y, y>5)$
 - $= wp(x:=x+1, wp(y:=x+y, y>5))$
 - $= wp(x:=x+1, x+y>5)$
 - $= x+1+y>5$
 - $= x+y>4$

Hoare Logic Rules

- Conditional
 - $\{ P \} \text{ if } x > 0 \text{ then } y := z \text{ else } y := -z \{ y > 5 \}$
 - What is the weakest precondition P ?

Hoare Logic Rules

- Conditional
 - $\{ P \} \text{ if } x > 0 \text{ then } y := z \text{ else } y := -z \{ y > 5 \}$
 - What is the weakest precondition P?
- Conditional rule
 - $wp(\text{if } B \text{ then } S \text{ else } T, Q) = B \Rightarrow wp(S, Q) \ \&\& \ \neg B \Rightarrow wp(T, Q)$
 - $wp(\text{if } x > 0 \text{ then } y := z \text{ else } y := -z, y > 5)$

Hoare Logic Rules

- Conditional

- $\{ P \} \text{ if } x > 0 \text{ then } y := z \text{ else } y := -z \{ y > 5 \}$
- What is the weakest precondition P ?

- Conditional rule

- $wp(\text{if } B \text{ then } S \text{ else } T, Q) = B \Rightarrow wp(S, Q) \ \&\& \ \neg B \Rightarrow wp(T, Q)$
- $wp(\text{if } x > 0 \text{ then } y := z \text{ else } y := -z, y > 5) = x > 0 \Rightarrow wp(y := z, y > 5) \ \&\& \ x \leq 0 \Rightarrow wp(y := -z, y > 5)$

Hoare Logic Rules

- Conditional

- $\{ P \} \text{ if } x > 0 \text{ then } y := z \text{ else } y := -z \{ y > 5 \}$
- What is the weakest precondition P?

- Conditional rule

- $wp(\text{if } B \text{ then } S \text{ else } T, Q) = B \Rightarrow wp(S, Q) \ \&\& \ \neg B \Rightarrow wp(T, Q)$
- $wp(\text{if } x > 0 \text{ then } y := z \text{ else } y := -z, y > 5) = x > 0 \Rightarrow wp(y := z, y > 5) \ \&\& \ x \leq 0 \Rightarrow wp(y := -z, y > 5)$

$$= x > 0 \Rightarrow z > 5 \ \&\& \ x \leq 0 \Rightarrow -z > 5$$

Hoare Logic Rules

- Conditional

- $\{ P \} \text{ if } x > 0 \text{ then } y := z \text{ else } y := -z \{ y > 5 \}$
- What is the weakest precondition P?

- Conditional rule

- $wp(\text{if } B \text{ then } S \text{ else } T, Q) = B \Rightarrow wp(S, Q) \ \&\& \ \neg B \Rightarrow wp(T, Q)$
- $wp(\text{if } x > 0 \text{ then } y := z \text{ else } y := -z, y > 5) = x > 0 \Rightarrow wp(y := z, y > 5) \ \&\& \ x \leq 0 \Rightarrow wp(y := -z, y > 5)$

$$= x > 0 \Rightarrow z > 5 \ \&\& \ x \leq 0 \Rightarrow -z > 5$$

$$= x > 0 \Rightarrow z > 5 \ \&\& \ x \leq 0 \Rightarrow z < -5$$

Hoare Logic Rules

- Loops
 - $\{ P \} \text{ while } (i < x) \text{ } f=f*i; i := i + 1 \{ f = x! \}$
 - What is the weakest precondition P ?

Hoare Logic Rules

- Loops
 - $\{ P \} \text{ while } (i < x) \text{ } f=f*i; i := i + 1 \{ f = x! \}$
 - What is the weakest precondition P ?
- Intuition
 - Must prove by induction
 - Only way to generalize across number of times loop executes
 - Need to guess induction hypothesis
 - Base case: precondition P
 - Inductive case: should be preserved by executing loop body

Proving loops correct

- First consider *partial correctness*
 - The loop may not terminate, but if it does, the postcondition will hold
- $\{P\} \text{ while } B \text{ do } S \{Q\}$
 - 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

Practice: Loop Invariants

Consider the following program:

```
{ N ≥ 0 }
```

```
i := 0;
```

```
while (i < N) do
```

```
    i := N
```

```
{ i = N }
```

Correctness Conditions

$P \Rightarrow \text{Inv}$

The invariant is initially true

$\{ \text{Inv} \ \&\& \ B \} \ S \ \{ \text{Inv} \}$

Loop preserves the invariant

$(\text{Inv} \ \&\& \ \neg B) \Rightarrow Q$

Invariant and exit implies
postcondition

Which of the following loop invariants are correct? For those that are incorrect, explain why.

A) $i = 0$

B) $i = N$

C) $N \geq 0$

D) $i \leq N$

Loop Example

- Prove array sum correct

$\{ N \geq 0 \}$

$j := 0;$

$s := 0;$

while ($j < N$) do

$j := j + 1;$

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

end

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

Loop Example

- Prove array sum correct

$\{ N \geq 0 \}$

$j := 0;$

$s := 0;$

while ($j < N$) do

$j := j + 1;$

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

end

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

How can we find a loop invariant?

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

end

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

Loop Example

- Prove array sum correct

$\{ N \geq 0 \}$

$j := 0;$

$s := 0;$

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

while ($j < N$) do

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

$j := j + 1;$

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

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

end

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

Loop Example

- Prove array sum correct

$\{ N \geq 0 \}$

$j := 0;$

$s := 0;$

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

while ($j < N$) do

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

$j := j + 1;$

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

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

end

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

Proof obligation #1

Proof obligation #2

Proof obligation #3

Proof Obligations

- Invariant is initially true

$\{ N \geq 0 \}$

$j := 0;$

$s := 0;$

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

Proof Obligations

- Invariant is initially true

$\{ N \geq 0 \}$

$j := 0;$

$s := 0;$

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

- Invariant is maintained

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

$j := j + 1;$

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

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

Proof Obligations

- Invariant is initially true

$\{ N \geq 0 \}$

$j := 0;$

$s := 0;$

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

- Invariant is maintained

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

$j := j + 1;$

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

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

- Invariant and exit condition imply postcondition

$0 \leq j \leq N \ \&\& \ s = (\sum i \mid 0 \leq i < j \cdot a[i]) \ \&\& \ j \geq N$

$\Rightarrow s = (\sum i \mid 0 \leq i < N \cdot a[i])$

Proof Obligations

- Invariant is initially true
 $\{ N \geq 0 \}$

$j := 0;$

$s := 0;$

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

Proof Obligations

- Invariant is initially true
 $\{ N \geq 0 \}$

```
j := 0;  
{  $0 \leq j \leq N$  &&  $0 = (\sum i \mid 0 \leq i < j \cdot a[i])$  } // by assignment rule  
s := 0;  
{  $0 \leq j \leq N$  &&  $s = (\sum i \mid 0 \leq i < j \cdot a[i])$  }
```

Proof Obligations

- Invariant is initially true

$\{ N \geq 0 \}$

$\{ 0 \leq \mathbf{0} \leq N \ \&\& \ 0 = (\sum i \mid 0 \leq i < \mathbf{0} \cdot a[i]) \}$ *// by assignment rule*

$j := 0;$

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

$s := 0;$

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

Proof Obligations

- Invariant is initially true

$\{ N \geq 0 \}$

$\{ 0 \leq \mathbf{0} \leq N \ \&\& \ 0 = (\sum i \mid 0 \leq i < \mathbf{0} \cdot a[i]) \}$ *// by assignment rule*

$j := 0;$

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

$s := 0;$

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

- Need to show that:

$(N \geq 0) \Rightarrow (0 \leq 0 \leq N \ \&\& \ 0 = (\sum i \mid 0 \leq i < 0 \cdot a[i]))$

Proof Obligations

- Invariant is initially true

$\{ N \geq 0 \}$

$\{ 0 \leq \mathbf{0} \leq N \ \&\& \ 0 = (\sum i \mid 0 \leq i < \mathbf{0} \cdot a[i]) \}$ *// by assignment rule*

$j := 0;$

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

$s := 0;$

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

- Need to show that:

$(N \geq 0) \Rightarrow (0 \leq 0 \leq N \ \&\& \ 0 = (\sum i \mid 0 \leq i < 0 \cdot a[i]))$

$= (N \geq 0) \Rightarrow (0 \leq N \ \&\& \ 0 = \mathbf{0})$ *// $0 \leq 0$ is true, empty sum is 0*

Proof Obligations

- Invariant is initially true

$\{ N \geq 0 \}$

$\{ 0 \leq \mathbf{0} \leq N \ \&\& \ 0 = (\sum i \mid 0 \leq i < \mathbf{0} \cdot a[i]) \}$ // by assignment rule

$j := 0;$

$\{ 0 \leq j \leq N \ \&\& \ \mathbf{0} = (\sum i \mid 0 \leq i < j \cdot a[i]) \}$ // by assignment rule

$s := 0;$

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

- Need to show that:

$(N \geq 0) \Rightarrow (0 \leq 0 \leq N \ \&\& \ 0 = (\sum i \mid 0 \leq i < 0 \cdot a[i]))$

$= (N \geq 0) \Rightarrow (0 \leq N \ \&\& \ 0 = \mathbf{0})$ // $0 \leq 0$ is true, empty sum is 0

$= (N \geq 0) \Rightarrow (0 \leq N)$ // $0=0$ is true, $P \ \&\& \ \text{true}$ is P

Proof Obligations

- Invariant is initially true

$\{ N \geq 0 \}$

$\{ 0 \leq \mathbf{0} \leq N \ \&\& \ 0 = (\sum i \mid 0 \leq i < \mathbf{0} \cdot a[i]) \}$ *// by assignment rule*

$j := 0;$

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

$s := 0;$

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

- Need to show that:

$(N \geq 0) \Rightarrow (0 \leq 0 \leq N \ \&\& \ 0 = (\sum i \mid 0 \leq i < 0 \cdot a[i]))$

$= (N \geq 0) \Rightarrow (0 \leq N \ \&\& \ 0 = \mathbf{0})$ *// $0 \leq 0$ is true, empty sum is 0*

$= (N \geq 0) \Rightarrow (0 \leq N)$ *// $0=0$ is true, $P \ \&\& \ \text{true}$ is P*

= true

Proof Obligations

- Invariant is maintained

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

$j := j + 1;$

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

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

Proof Obligations

- Invariant is maintained

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

$j := j + 1;$

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

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

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

Proof Obligations

- Invariant is maintained

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

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

$j := j + 1;$

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

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

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

Proof Obligations

- Invariant is maintained

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

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

$j := j + 1;$

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

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

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

- Need to show that:

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

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

Proof Obligations

- Invariant is maintained

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

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

$j := j + 1;$

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

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

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

- Need to show that:

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

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

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

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

Proof Obligations

- Invariant is maintained

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

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

$j := j + 1;$

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

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

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

- Need to show that:

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

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

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

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

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

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

Proof Obligations

- Invariant is maintained

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

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

$j := j + 1;$

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

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

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

- Need to show that:

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

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

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

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

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

$\Rightarrow \mathbf{(-1 \leq j < N \ \&\& \ s + a[j+1] = (\sum_i \mid 0 \leq i < j \cdot a[i]) + a[j])}$ *// 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

$\{ N \geq 0 \}$

$j := 0;$

$s := 0;$

while ($j < N$) do

$j := j + 1;$

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

end

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

Where's the error?

- Prove array sum correct

$\{ N \geq 0 \}$

$j := 0;$

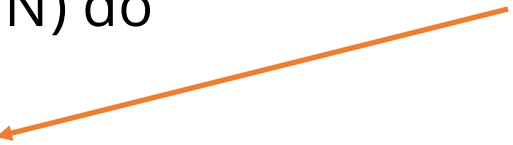
$s := 0;$

while ($j < N$) do

$j := j + 1;$

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

Need to add element
before incrementing j



end

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

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

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

Proof Obligations

- Invariant is maintained

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

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

$j := j + 1;$

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

Proof Obligations

- Invariant is maintained

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

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

$\{0 \leq \mathbf{j+1} \leq N \ \&\& \ s = (\sum i \mid 0 \leq i < \mathbf{j+1} \cdot a[i]) \}$

$j := j + 1;$

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

// by assignment rule

Proof Obligations

- Invariant is maintained

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

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

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

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

$j := j + 1;$

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

Proof Obligations

- Invariant is maintained

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

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

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

$\{0 \leq j+1 \leq N \ \&\& \ s = (\sum i \mid 0 \leq i < j+1 \cdot a[i]) \}$

// by assignment rule

$j := j + 1;$

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

- Need to show that:

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

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

Proof Obligations

- Invariant is maintained

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

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

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

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

$j := j + 1;$

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

- Need to show that:

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

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

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

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

Proof Obligations

- Invariant is maintained

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

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

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

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

$j := j + 1;$

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

- Need to show that:

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

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

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

$\Rightarrow (-1 \leq j < N \ \&\& \ s+a[j] = (\sum i \mid 0 \leq i < j+1 \cdot a[i]))$ // simplify bounds of j

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

$\Rightarrow (-1 \leq j < N \ \&\& \ s+a[j] = (\sum i \mid 0 \leq i < j \cdot a[i]) + a[j])$ // separate last part of sum

Proof Obligations

- Invariant is maintained

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

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

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

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

$j := j + 1;$

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

- Need to show that:

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

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

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

$\Rightarrow (-1 \leq j < N \ \&\& \ s+a[j] = (\sum_i \mid 0 \leq i < j+1 \cdot a[i]))$ *// simplify bounds of j*

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

$\Rightarrow (-1 \leq j < N \ \&\& \ s+a[j] = (\sum_i \mid 0 \leq i < j \cdot a[i]) + a[j])$ *// separate last part of sum*

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

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

Proof Obligations

- Invariant is maintained

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

$\{0 \leq j+1 \leq N \ \&\& \ s+a[j] = (\sum i \mid 0 \leq i < j+1 \cdot a[i]) \}$

// by assignment rule

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

$\{0 \leq j+1 \leq N \ \&\& \ s = (\sum i \mid 0 \leq i < j+1 \cdot a[i]) \}$

// by assignment rule

$j := j + 1;$

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

- Need to show that:

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

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

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

$\Rightarrow (-1 \leq j < N \ \&\& \ s+a[j] = (\sum i \mid 0 \leq i < j+1 \cdot a[i]))$ *// simplify bounds of j*

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

$\Rightarrow (-1 \leq j < N \ \&\& \ s+a[j] = (\sum i \mid 0 \leq i < j \cdot a[i]) + a[j])$ *// separate last part of sum*

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

$\Rightarrow (-1 \leq j < N \ \&\& \ s = (\sum i \mid 0 \leq i < j \cdot a[i]))$

// subtract a[j] from both sides

$= \text{true}$

// $0 \leq j \Rightarrow -1 \leq j$

Proof Obligations

- Invariant and exit condition implies postcondition

$$0 \leq j \leq N \ \&\& \ s = (\sum i \mid 0 \leq i < j \cdot a[i]) \ \&\& \ j \geq N$$

$$\Rightarrow s = (\sum i \mid 0 \leq i < N \cdot a[i])$$

Proof Obligations

- Invariant and exit condition implies postcondition

$$0 \leq j \leq N \ \&\& \ s = (\sum i \mid 0 \leq i < j \cdot a[i]) \ \&\& \ j \geq N$$

$$\Rightarrow s = (\sum i \mid 0 \leq i < N \cdot a[i])$$

$$= 0 \leq j \ \&\& \ \mathbf{j = N} \ \&\& \ s = (\sum i \mid 0 \leq i < j \cdot a[i])$$

$$\Rightarrow s = (\sum i \mid 0 \leq i < N \cdot a[i])$$

$$// \textit{because } (j \leq N \ \&\& \ j \geq N) = (j = N)$$

Proof Obligations

- Invariant and exit condition implies postcondition

$$0 \leq j \leq N \ \&\& \ s = (\sum i \mid 0 \leq i < j \cdot a[i]) \ \&\& \ j \geq N$$

$$\Rightarrow s = (\sum i \mid 0 \leq i < N \cdot a[i])$$

$$= 0 \leq j \ \&\& \ \mathbf{j} = \mathbf{N} \ \&\& \ s = (\sum i \mid 0 \leq i < j \cdot a[i])$$

$$\Rightarrow s = (\sum i \mid 0 \leq i < N \cdot a[i])$$

$$\text{// because } (j \leq N \ \&\& \ j \geq N) = (j = N)$$

$$= 0 \leq \mathbf{N} \ \&\& \ s = (\sum i \mid 0 \leq i < \mathbf{N} \cdot a[i]) \Rightarrow s = (\sum i \mid 0 \leq i < N \cdot a[i])$$

$$\text{// by substituting } N \text{ for } j, \text{ since } j = N$$

Proof Obligations

- Invariant and exit condition implies postcondition

$$0 \leq j \leq N \ \&\& \ s = (\sum i \mid 0 \leq i < j \cdot a[i]) \ \&\& \ j \geq N$$

$$\Rightarrow s = (\sum i \mid 0 \leq i < N \cdot a[i])$$

$$= 0 \leq j \ \&\& \ \mathbf{j = N} \ \&\& \ s = (\sum i \mid 0 \leq i < j \cdot a[i])$$

$$\Rightarrow s = (\sum i \mid 0 \leq i < N \cdot a[i])$$

$$\text{// because } (j \leq N \ \&\& \ j \geq N) = (j = N)$$

$$= 0 \leq \mathbf{N} \ \&\& \ s = (\sum i \mid 0 \leq i < \mathbf{N} \cdot a[i]) \Rightarrow s = (\sum i \mid 0 \leq i < N \cdot a[i])$$

$$\text{// by substituting } N \text{ for } j, \text{ since } j = N$$

$$= \mathbf{true} \text{// because } P \ \&\& \ Q \Rightarrow Q$$

Practice: Writing Proof Obligations

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

$\{ N \geq 0 \}$

$i := 0;$

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

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

$VCGen(\text{skip}, Q) =$

$VCGen(S_1; S_2, Q) =$

$VCGen(\text{if } b \text{ then } S_1 \text{ else } S_2, Q) =$

$VCGen(x := e, Q) =$

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

$$VCGen(\text{skip}, Q) = Q$$

$$VCGen(S_1; S_2, Q) = VCGen(S_1, VCGen(S_2, Q))$$

$$VCGen(\text{if } b \text{ then } S_1 \text{ else } S_2, Q) = b \Rightarrow VCGen(S_1, Q) \wedge \neg b \Rightarrow VCGen(S_2, Q)$$

$$VCGen(x := e, Q) = [e/x]Q$$

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

Loops are special — as usual:

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.