Lecture 12:
Axiomatic Semantics and Hoare Logic

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

Rohan Padhye and Jonathan Aldrich

March 16, 2021

* Course materials developed with Claire Le Goues

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.

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 $<S;\ E> \Downarrow E'$, 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 Language

$$\begin{array}{lllllll} A & ::= & \text{true} & \mid \text{false} & \mid e_1 = e_2 & \mid e_1 \geq e_2 & \mid A_1 \wedge A_2 \\ & \mid & A_1 \vee A_2 & \mid A_1 \Rightarrow A_2 & \mid \forall x.A & \mid \exists x.A & \end{array}$$

- x,y quantify over integers
  - We will be slightly sloppy and mix logical and program variables
- We'll treat Boolean expressions as a special case of assertions

Assertion Semantics

- $E \models A$ means A is true in E
- Rules: $E \models \text{true}$

  $E \models e_1 = e_2$

  $E \models e_1 \geqslant e_2$

  $E \models A_1 \wedge A_2$

  $\ldots$

  $E \models \forall x.A$

  $E \models \exists x.A$

Assertion Semantics

- $E \models A$ means A is true in E
- Rules:

$$
\begin{array}{ll}
E \models \text{true} & \textit{always} \\
E \models e_1 = e_2 & \textit{iff } \langle E, e_1 \rangle \Downarrow n \text{ and } \langle E, e_2 \rangle \Downarrow n \\
E \models e_1 \geqslant e_2 & \textit{iff } \langle E, e_1 \rangle \Downarrow n_1, \langle E, e_2 \rangle \Downarrow n_2, \text{ and } n_1 \geqslant n_2 \\
E \models A_1 \wedge A_2 & \textit{iff } E \models A_1 \text{ and } E \models A_2 \\
\ldots & \\
E \models \forall x. A & \textit{iff } \forall n \in \mathbb{Z}. E[x \mapsto n] \models A \\
E \models \exists x. A & \textit{iff } \exists n \in \mathbb{Z}. E[x \mapsto n] \models A
\end{array}
$$

Practice: Exploring Hoare Triples

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

{ true }	x := 5	{ }
{ }	x : = x + 3	{ x = y + 3 }
{ }	x := x * 2 + 3	{ x > 1 }
{ x = a }	if (x < 0) then x := -x	{ }
{ false }	x := 3	{ }
{ x < 0 }	while (x != 0) x := x – 1	{ }

Strongest Postconditions

- Here are a number of valid Hoare Triples:
  - {x = 5} x := x * 2 { true }
  - {x = 5} x := x * 2 { x > 0 }
  - {x = 5} x := x * 2 { x = 10 || 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 { true }
  - {x = 5} x := x * 2 { x > 0 }
  - {x = 5} x := x * 2 { x = 10 || 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 $\Rightarrow$ true
  - check: x = 10 $\Rightarrow$ x > 0
  - check: x = 10 $\Rightarrow$ x = 10 || x = 5
  - check: x = 10 $\Rightarrow$ x = 10

Weakest Preconditions

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

Weakest Preconditions

- Here are a number of valid Hoare Triples:
  - {x = 5 && y = 10} z := x / y { z < 1 }
  - {x < y && y > 0} z := x / y { z < 1 }
  - {y ≠ 0 && x / y < 1} z := x / y { 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 ≠ 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

- {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

- Question: Could we state a similar theorem for a strongest postcondition function?
  - e.g. {P} S {Q} holds if and only if $sp(S,P) \Rightarrow Q$

Hoare Triples and Weakest Preconditions

- {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

- 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

Practice: 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) { false } x := 2 / y { 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*y + z { x * y - z > 0 }
  - What is the weakest precondition 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 }

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

Hoare Logic Rules

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

Hoare Logic Rules

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

Hoare Logic Rules

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

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

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

Practice: Preconditions/Postconditions

Fill in the missing pre- or post-conditions with predicates that make each Hoare triple valid.

A) { x = y } x := y * 2 {   }

B) {   } x := x + 3 { x = z }

C) {   } x := x + 1; y := y * x { y = 2 * z }

D) {   } if (x > 0) then y := x else y := 0 { y > 0 }

Hoare Logic Rules

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

Hoare Logic Rules

- Loops
  - { P } while (i < x) 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} 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

Practice: Loop Invariants

Consider the following program:

```
{ N >= 0 }
i := 0;
while (i < N) do
  i := N
{ i = N }
```

Correctness Conditions

$P \Rightarrow Inv$

The invariant is initially true

{ Inv && B } S {Inv}

Loop preserves the invariant

$(Inv \;\&\&\; \neg B) \Rightarrow Q$

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 >= 0

D) i <= 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
```

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

How can we find a loop invariant?

Replace N with j

Add information on range of j

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

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; Proof obligation #1

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

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; Proof obligation #2

  s := s + a[j];

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

end Proof obligation #3

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

Proof Obligations

- Invariant is initially true

  { N ≥ 0 }

  j := 0;

  s := 0;

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

Proof Obligations

- Invariant is initially true

  $\{ N \geq 0 \}$

  j := 0;

  s := 0;

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

- Invariant is maintained

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

  j := j + 1;

  s := s + a[j];

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

Proof Obligations

- Invariant is initially true

  $\{ N \geq 0 \}$

  j := 0;

  s := 0;

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

- Invariant is maintained

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

  j := j + 1;

  s := s + a[j];

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

- Invariant and exit condition imply postcondition

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

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

Proof Obligations

- Invariant is initially true

$\{ N \geq 0 \}$

j := 0;

s := 0;

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

Proof Obligations

- Invariant is initially true

  $\{ N \geq 0 \}$

  j := 0;

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

  s := 0;

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

Proof Obligations

- Invariant is initially true

{ N ≥ 0 }

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

j := 0;

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

s := 0;

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

Proof Obligations

- Invariant is initially true

  $\{ N \geq 0 \}$

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

  j := 0;

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

  s := 0;

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

- Need to show that:

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

Proof Obligations

- Invariant is initially true

  $\{ N \geq 0 \}$

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

  j := 0;

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

  s := 0;

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

- Need to show that:

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

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

Proof Obligations

- Invariant is initially true

  $\{ N \geq 0 \}$

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

  j := 0;

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

  s := 0;

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

- Need to show that:

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

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

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

Proof Obligations

- Invariant is initially true

  $\{ N \geq 0 \}$

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

  j := 0;

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

  s := 0;

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

- Need to show that:

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

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

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

= true