

CS720

Logical Foundations of Computer Science

Lecture 15: Program verification (part 2)

Tiago Cogumbreiro

Why are we learning this?

In this class we are learning about three techniques:

- **formalize the PL semantics** (eg, formalize an imperative PL)
- **prove PL properties** (eg, composing Hoare triples)
- **verify programs** (eg, proving that an algorithm follows a given specification)

Summary

- Consequence Theorem
- Conditional Theorem
- While-Loop Theorem
- Axiomatic Hoare Logic

Theorems help us structure our proofs

```
Goal {{ (fun st : state => st X = 2) [X |→ X + 1] [ X |→ 1] }}  
  X ::= 1;; X ::= X + 1  
  {{ fun st => st X = 2 }}.
```

Two alternative proofs

Proof.

```
  apply hoare_seq  
    with (Q:=(fun st => st X=2)[X |→ X+1]). {  
      apply hoare_asgn.  
    }  
  apply hoare_asgn.
```

Qed.

Proof.

```
  unfold hoare_triple.  
  intros st_in st_out runs H_holds.  
  invc runs.  
  invc H1.  
  invc H4.  
  reflexivity.
```

Qed.

What if the pre- does not match H-asgn?

```
Goal {{ fun st => True }}  
  X := 1; X := X + 1  
  {{ fun st => st X = 2 }}.
```

What if the pre- does not match H-asgn?

```
Goal {{ fun st => True }}  
  X := 1; X := X + 1  
  {{ fun st => st X = 2 }}.
```

- **Provable, but not using H-asgn and H-seq.**

Let us build a theory on assertions

Assertion implication/equivalence

- Define A **implies** assertion B , notation $A \rightarrow B$, if, and only if, for any state s ,
 $A(s) \implies B(s)$.
 - Define assertion equivalence between A and B , notation $A \leftrightarrow B$, if, and only if,
 $A(s) \iff B(s)$ for any state s .
1. $\{x = 3\} \rightarrow \{x = 3 \vee x \leq y\}$

Assertion implication/equivalence

- Define A **implies** assertion B , notation $A \rightarrow B$, if, and only if, for any state s ,
 $A(s) \implies B(s)$.
 - Define assertion equivalence between A and B , notation $A \leftrightarrow B$, if, and only if,
 $A(s) \iff B(s)$ for any state s .
1. $\{x = 3\} \rightarrow \{x = 3 \vee x \leq y\}$
 2. $\{x \neq x\} \rightarrow \{x = 3\}$

Assertion implication/equivalence

- Define A **implies** assertion B , notation $A \rightarrow B$, if, and only if, for any state s ,
 $A(s) \implies B(s)$.
 - Define assertion equivalence between A and B , notation $A \iff B$, if, and only if,
 $A(s) \iff B(s)$ for any state s .
1. $\{x = 3\} \rightarrow \{x = 3 \vee x \leq y\}$
 2. $\{x \neq x\} \rightarrow \{x = 3\}$
 3. $\{x \leq y\} \iff \{x < y \vee x = y\}$

Assertion implication/equivalence

- Define A **implies** assertion B , notation $A \rightarrow B$, if, and only if, for any state s ,
 $A(s) \implies B(s)$.
 - Define assertion equivalence between A and B , notation $A \iff B$, if, and only if,
 $A(s) \iff B(s)$ for any state s .
1. $\{x = 3\} \rightarrow \{x = 3 \vee x \leq y\}$
 2. $\{x \neq x\} \rightarrow \{x = 3\}$
 3. $\{x \leq y\} \iff \{x < y \vee x = y\}$
 4. $\{x = 2[x \mapsto x + 1][x \mapsto 1]\} \iff \{\top\}$

Assertion implication/equivalence

- Define A **implies** assertion B , notation $A \rightarrow B$, if, and only if, for any state s ,
 $A(s) \implies B(s)$.
- Define assertion equivalence between A and B , notation $A \iff B$, if, and only if,
 $A(s) \iff B(s)$ for any state s .

1. $\{x = 3\} \rightarrow \{x = 3 \vee x \leq y\}$
2. $\{x \neq x\} \rightarrow \{x = 3\}$
3. $\{x \leq y\} \iff \{x < y \vee x = y\}$
4. $\{x = 2[x \mapsto x + 1][x \mapsto 1]\} \iff \{\top\}$

Goal `((fun st => st X = 2) [X |> X + 1] [X |> 1]) <=> (fun st => True).`

Proof.

```
unfold assn_sub, assert_implies; auto.
```

Qed.



Weakening and strengthening pre-/post conditions

We know that $\{\top\} x := 1; x := x + 1 \{x = 2\}$ holds.

1. $\{y = 1\} x := 1; x := x + 1 \{x = 2\}$

Weakening and strengthening pre-/post conditions

We know that $\{\top\} x := 1; x := x + 1 \{x = 2\}$ holds.

1. $\{\mathbf{y = 1}\} x := 1; x := x + 1 \{x = 2\}$ **Holds.**

Strengthen pre-condition: $\{\mathbf{y = 1}\} \rightarrow \{\top\}$

Weakening and strengthening pre-/post conditions

We know that $\{\top\} x := 1; x := x + 1 \{x = 2\}$ holds.

1. $\{\mathbf{y = 1}\} x := 1; x := x + 1 \{x = 2\}$ **Holds.**

Strengthen pre-condition: $\{\mathbf{y = 1}\} \rightarrow \{\top\}$

2. $\{\mathbf{x = 10}\} x := 1; x := x + 1 \{x = 2\}$

Weakening and strengthening pre-/post conditions

We know that $\{\top\} x := 1; x := x + 1 \{x = 2\}$ holds.

1. $\{\mathbf{y = 1}\} x := 1; x := x + 1 \{x = 2\}$ **Holds.**

Strengthen pre-condition: $\{\mathbf{y = 1}\} \rightarrow \{\top\}$

2. $\{\mathbf{x = 10}\} x := 1; x := x + 1 \{x = 2\}$ **Holds.**

Strengthen pre-condition: $\{\mathbf{x = 10}\} \rightarrow \{\top\}$

Weakening and strengthening pre-/post conditions

We know that $\{\top\} x := 1; x := x + 1 \{x = 2\}$ holds.

1. $\{\mathbf{y = 1}\} x := 1; x := x + 1 \{x = 2\}$ **Holds.**

Strengthen pre-condition: $\{\mathbf{y = 1}\} \rightarrow \{\top\}$

2. $\{\mathbf{x = 10}\} x := 1; x := x + 1 \{x = 2\}$ **Holds.**

Strengthen pre-condition: $\{\mathbf{x = 10}\} \rightarrow \{\top\}$

3. $\{\top\} x := 1; x := x + 1 \{x = 2 \wedge \mathbf{y = 1}\}$

Weakening and strengthening pre-/post conditions

We know that $\{\top\} x := 1; x := x + 1 \{x = 2\}$ holds.

1. $\{\mathbf{y = 1}\} x := 1; x := x + 1 \{x = 2\}$ **Holds.**

Strengthen pre-condition: $\{\mathbf{y = 1}\} \rightarrow \{\top\}$

2. $\{\mathbf{x = 10}\} x := 1; x := x + 1 \{x = 2\}$ **Holds.**

Strengthen pre-condition: $\{\mathbf{x = 10}\} \rightarrow \{\top\}$

3. $\{\top\} x := 1; x := x + 1 \{x = 2 \wedge \mathbf{y = 1}\}$ **Does NOT hold.**

Strengthen post-condition: $\{x = 2 \wedge \mathbf{y = 1}\} \rightarrow \{x = 2\}$

Weakening and strengthening pre-/post conditions

We know that $\{\top\} x := 1; x := x + 1 \{x = 2\}$ holds.

1. $\{\mathbf{y = 1}\} x := 1; x := x + 1 \{x = 2\}$ **Holds.**

Strengthen pre-condition: $\{\mathbf{y = 1}\} \rightarrow \{\top\}$

2. $\{\mathbf{x = 10}\} x := 1; x := x + 1 \{x = 2\}$ **Holds.**

Strengthen pre-condition: $\{\mathbf{x = 10}\} \rightarrow \{\top\}$

3. $\{\top\} x := 1; x := x + 1 \{x = 2 \wedge \mathbf{y = 1}\}$ **Does NOT hold.**

Strengthen post-condition: $\{x = 2 \wedge \mathbf{y = 1}\} \rightarrow \{x = 2\}$

4. $\{\top\} x := 1; x := x + 1 \{\top\}$

Weakening and strengthening pre-/post conditions

We know that $\{\top\} x := 1; x := x + 1 \{x = 2\}$ holds.

1. $\{\mathbf{y = 1}\} x := 1; x := x + 1 \{x = 2\}$ **Holds.**

Strengthen pre-condition: $\{\mathbf{y = 1}\} \rightarrow \{\top\}$

2. $\{\mathbf{x = 10}\} x := 1; x := x + 1 \{x = 2\}$ **Holds.**

Strengthen pre-condition: $\{\mathbf{x = 10}\} \rightarrow \{\top\}$

3. $\{\top\} x := 1; x := x + 1 \{x = 2 \wedge \mathbf{y = 1}\}$ **Does NOT hold.**

Strengthen post-condition: $\{x = 2 \wedge \mathbf{y = 1}\} \rightarrow \{x = 2\}$

4. $\{\top\} x := 1; x := x + 1 \{\top\}$ **Holds.**

Weaken post-condition: $\{x = 2\} \rightarrow \{\top\}$

Weakening and strengthening pre-/post conditions

We know that $\{\top\} x := 1; x := x + 1 \{x = 2\}$ holds.

1. $\{\mathbf{y = 1}\} x := 1; x := x + 1 \{x = 2\}$ **Holds.**

Strengthen pre-condition: $\{\mathbf{y = 1}\} \rightarrow \{\top\}$

2. $\{\mathbf{x = 10}\} x := 1; x := x + 1 \{x = 2\}$ **Holds.**

Strengthen pre-condition: $\{\mathbf{x = 10}\} \rightarrow \{\top\}$

3. $\{\top\} x := 1; x := x + 1 \{x = 2 \wedge \mathbf{y = 1}\}$ **Does NOT hold.**

Strengthen post-condition: $\{x = 2 \wedge \mathbf{y = 1}\} \rightarrow \{x = 2\}$

4. $\{\top\} x := 1; x := x + 1 \{\top\}$ **Holds.**

Weaken post-condition: $\{x = 2\} \rightarrow \{\top\}$

5. $\{\top\} x := 1; x := x + 1 \{\perp\}$

Weakening and strengthening pre-/post conditions

We know that $\{\top\} x := 1; x := x + 1 \{x = 2\}$ holds.

1. $\{\mathbf{y = 1}\} x := 1; x := x + 1 \{x = 2\}$ **Holds.**

Strengthen pre-condition: $\{\mathbf{y = 1}\} \rightarrow \{\top\}$

2. $\{\mathbf{x = 10}\} x := 1; x := x + 1 \{x = 2\}$ **Holds.**

Strengthen pre-condition: $\{\mathbf{x = 10}\} \rightarrow \{\top\}$

3. $\{\top\} x := 1; x := x + 1 \{x = 2 \wedge \mathbf{y = 1}\}$ **Does NOT hold.**

Strengthen post-condition: $\{x = 2 \wedge \mathbf{y = 1}\} \rightarrow \{x = 2\}$

4. $\{\top\} x := 1; x := x + 1 \{\top\}$ **Holds.**

Weaken post-condition: $\{x = 2\} \rightarrow \{\top\}$

5. $\{\top\} x := 1; x := x + 1 \{\perp\}$ **Does NOT hold.**

Strengthen post-condition: $\{\perp\} \rightarrow \{x = 2\}$

Proving H-cons

Theorem hoare_consequence_pre : forall (P P' Q : Assertion) c,
 {{P'}} c {{Q}} →
 P → P' →
 {{P}} c {{Q}}.

Theorem hoare_consequence_post : forall (P Q Q' : Assertion) c,
 {{P}} c {{Q'}} →
 Q' → Q →
 {{P}} c {{Q}}.

Theorem hoare_consequence : forall (P P' Q Q' : Assertion) c,
 {{P'}} c {{Q'}} →
 P → P' →
 Q' → Q →
 {{P}} c {{Q}}.

Exercise

```
Goal {{fun st => True}}  
  X := 1; X := X + 1  
  {{fun st => st X = 2}}.
```

Conditionals

Theorem (H-cond): If $\{P\} c_1 \{Q\}$ and $\{P\} c_2 \{Q\}$, then $\{P\} \text{if } b \text{ then } c_1 \text{ else } c_2 \{Q\}$.

Theorem hoare_cond: forall P Q b c1 c2,
 $\{\{P\}\} c_1 \{\{Q\}\} \rightarrow$
 $\{\{P\}\} c_2 \{\{Q\}\} \rightarrow$
 $\{\{P\}\} \text{if } b \text{ then } c_1 \text{ else } c_2 \{\{Q\}\}.$

Prove that

$$\frac{\{ \top \} y := 2 \{ x \leq y \} \quad \{ \top \} y := x + 1 \{ x \leq y \}}{\{ \top \} \text{if } x = 0 \text{ then } y := 2 \text{ else } y := x + 1 \{ x \leq y \}} \text{H-cond}$$

Conditionals

Proving **else**:

$$\frac{\frac{\dots}{\{T\} \rightarrow \{x \leq y[y \mapsto x + 1]\}} \quad \frac{\dots}{\{x \leq y[y \mapsto x + 1]\}y ::= x + 1\{x \leq y\}} \text{H-asgn}}{\{T\}y ::= x + 1\{x \leq y\}} \text{H-cons-pre}}{\{T\}\text{if } x = 0 \text{ then } y ::= 2 \text{ else } y ::= x + 1 \{x \leq y\}} \text{H-cond}$$

Conditionals

Proving **else**:

$$\frac{\frac{\dots}{\{T\} \rightarrow \{x \leq y[y \mapsto x + 1]\}} \quad \frac{\dots}{\{x \leq y[y \mapsto x + 1]\}y ::= x + 1\{x \leq y\}} \text{H-asgn}}{\{T\}y ::= x + 1\{x \leq y\}} \text{H-cons-pre}}{\{T\}\text{if } x = 0 \text{ then } y ::= 2 \text{ else } y ::= x + 1 \{x \leq y\}} \text{H-cond}$$

Proving **then**:

$$\frac{\frac{\text{???}}{\{T\}y ::= 2\{x \leq y\}}}{\{T\}\text{if } x = 0 \text{ then } y := 2 \text{ else } y := x + 1 \{x \leq y\}} \text{H-cond}$$

Conditionals

Proving **else**:

$$\frac{\frac{\dots}{\{T\} \rightarrow \{x \leq y[y \mapsto x + 1]\}} \quad \frac{\dots}{\{x \leq y[y \mapsto x + 1]\}y ::= x + 1\{x \leq y\}} \text{H-asgn}}{\{T\}y ::= x + 1\{x \leq y\}} \text{H-cons-pre}}{\{T\}\text{if } x = 0 \text{ then } y ::= 2 \text{ else } y ::= x + 1 \{x \leq y\}} \text{H-cond}$$

Proving **then**:

$$\frac{\frac{???}{\{T\}y ::= 2\{x \leq y\}}}{\{T\}\text{if } x = 0 \text{ then } y := 2 \text{ else } y := x + 1 \{x \leq y\}} \text{H-cond}$$

We are missing that $x = 0$, which would help us prove this result!

The Hoare theorem for If

Theorem (H-if): If $\{P \wedge b\} c_1 \{Q\}$ and $\{P \wedge \neg b\} c_2 \{Q\}$, then $\{P\} \text{if } b \text{ then } c_1 \text{ else } c_2 \{Q\}$.

The Hoare theorem for If in Coq

Definition `bassn b : Assertion := fun st => (beval st b = true).`

Theorem `hoare_if : forall P Q b c1 c2,`
 `{{fun st => P st /\ bassn b st}} c1 {{Q}} ->`
 `{{fun st => P st /\ ~(bassn b st)}} c2 {{Q}} ->`
 `{{P}} (if b then c1 else c2 FI) {{Q}}.`

Proof.

`intros.`

Example

Goal

```
{{fun st  $\Rightarrow$  True}}  
if X = 0  
then Y := 2  
else Y := X + 1  
{{fun st  $\Rightarrow$  st X  $\leq$  st Y}}.
```

The Hoare theorem for While

1. $\{P\} \text{ while } b \text{ do } c \text{ end } \{P\}$

The Hoare theorem for While

1. $\{P\} \text{ while } b \text{ do } c \text{ end } \{P\}$
2. $\{P\} \text{ while } b \text{ do } c \text{ end } \{P \wedge \neg b\}$

We know that b is false after the loop. Can we state something about the body of the loop?

The Hoare theorem for While

1. $\{P\} \text{ while } b \text{ do } c \text{ end } \{P\}$
2. $\{P\} \text{ while } b \text{ do } c \text{ end } \{P \wedge \neg b\}$

We know that b is false after the loop. Can we state something about the body of the loop?

3. If $\{P\} c \{P\}$, then $\{P\} \text{ while } b \text{ do } c \text{ end } \{P \wedge \neg b\}$

We know that the loop body must at least preserve $\{P\}$. Why? Can we do better?

The Hoare theorem for While

1. $\{P\} \text{ while } b \text{ do } c \text{ end } \{P\}$
2. $\{P\} \text{ while } b \text{ do } c \text{ end } \{P \wedge \neg b\}$

We know that b is false after the loop. Can we state something about the body of the loop?

3. If $\{P\} c \{P\}$, then $\{P\} \text{ while } b \text{ do } c \text{ end } \{P \wedge \neg b\}$

We know that the loop body must at least preserve $\{P\}$. Why? Can we do better?

Theorem (H-while): If $\{P \wedge b\} c \{P\}$, then $\{P\} \text{ while } b \text{ do } c \text{ end } \{P \wedge \neg b\}$.

```
Theorem hoare_while : forall P b c,  
  {{{fun st => P st /\ bassn b st}}} c {{{P}}} ->  
  {{{P}}} while b do c end {{{fun st => P st /\ ~ (bassn b st)}}}.
```

Proof.

```
unfold hoare_triple; intros.
```



Example

```
Example while_example :  
  {{fun st  $\Rightarrow$  st X  $\leq$  3}}  
  while X  $\leq$  2  
  do X := X + 1 end  
  {{fun st  $\Rightarrow$  st X = 3}}.  
Proof.
```

Recap

- We introduced Hoare triples $\{P\} c \{Q\}$ as a framework to specify programs
- We introduced a set of theorems (syntax-oriented) to help us prove results on Hoare triples.

Hoare Logic Theory

$\{P\} \text{ skip } \{P\}$ (H-skip) $\{P[x \mapsto a]\} x ::= a \{P\}$ (H-asgn)

$$\frac{\{P\} c_1 \{Q\} \quad \{Q\} c_2 \{R\}}{\{P\} c_1; c_2 \{R\}} \text{ (H-seq)}$$
$$\frac{P \twoheadrightarrow P' \quad \{P'\} c \{Q'\} \quad Q' \twoheadrightarrow Q}{\{P\} c \{Q\}} \text{ (H-cons)}$$
$$\frac{\{P \wedge b\} c_1 \{Q\} \quad \{P \wedge \neg b\} c_2 \{Q\}}{\{P\} \text{ if } b \text{ then } c_1 \text{ else } c_2 \{Q\}} \text{ (H-if)}$$
$$\frac{\{P \wedge b\} c \{P\}}{\{P\} \text{ while } b \text{ do } c \text{ end } \{P \wedge \neg b\}} \text{ (H-while)}$$

Hoare Logic as an Axiomatic Logic

- The set of theorems in slide 12 can describe Hoare's Logic **axiomatically**
- **Necessary** condition (sound): $\text{hoare_proof}(P, c, Q) \rightarrow \{P\} c \{Q\}$
- **Sufficient** condition (complete): $\{P\} c \{Q\} \rightarrow \text{hoare_proof}(P, c, Q)$

```
Inductive hoare_proof : Assertion → com → Assertion → Type :=
| H_Skip : forall P, hoare_proof P (SKIP) P
| H_Asgn  : forall Q V a, hoare_proof (assn_sub V a Q) (V ::= a) Q
| H_Seq   : forall P c Q d R, hoare_proof P c Q → hoare_proof Q d R → hoare_proof P (c;;d) R
| H_If    : forall P Q b c1 c2,
  hoare_proof (fun st ⇒ P st ∧ bassn b st) c1 Q →
  hoare_proof (fun st ⇒ P st ∧ ~(bassn b st)) c2 Q →
  hoare_proof P (IFB b THEN c1 ELSE c2 FI) Q
| H_While : forall P b c,
  hoare_proof (fun st ⇒ P st ∧ bassn b st) c P →
  hoare_proof P (WHILE b DO c END) (fun st ⇒ P st ∧ ~ (bassn b st))
| H_Consequence : forall (P Q P' Q' : Assertion) c,
  hoare_proof P' c Q' → (forall st, P st → P' st) → (forall st, Q' st → Q st) → hoare_proof P c Q.
```

Summary

- Consequence Theorem
- Conditional Theorem
- While-Loop Theorem
- Axiomatic Hoare Logic