CS720

Logical Foundations of Computer Science

Lecture 15: Program verification (part 2)

Tiago Cogumbreiro

`Equiv.v`

Due Thursday October 25, 11:59pm EST

`Imp.v`

Due Friday October 26, 11:59pm EST

`Hoare.v`

Due Thursday November 1, 11:59pm EST

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

Exercise

Does $\{x = 2[x \mapsto x + 1][x \mapsto 1]\}\ x ::= 1; ; x ::= x + 1\ \{x = 2\}$ hold?

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

Exercise

> Does $\{x = 2[x \mapsto x+1][x \mapsto 1]\}\ x ::= 1;;x ::= x+1\ \{x = 2\}$ hold?

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

> Yes. Does $\{\top\}\ x ::= 1;;x ::= x+1\ \{x = 2\}$ hold? And, can we prove it T-seq and T-asgn?

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

Exercise

> Does $\{x = 2[x \mapsto x+1][x \mapsto 1]\}\ x ::= 1;;x ::= x + 1\ \{x = 2\}$ hold?

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

> Yes. Does $\{\top\}\ x ::= 1;;x ::= x + 1\ \{x = 2\}$ hold? And, can we prove it T-seq and T-asgn?

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

> **No.** The pre-condition has to match what we stated H-asgn. But we know that the above statement holds. Let us write a new theorem that handles such cases.

Assertion implication

We say that assertion A *implies* assertion B, notation $A \twoheadrightarrow B$, if, and only if, for any state s, $A(s) \implies B(s)$. Similarly, we say that two assertions are equivalent, notation $A \twoheadleftarrow\!\!\twoheadrightarrow B$, if, and only if, $A(s) \iff B(s)$ for any state s.

1. $\{x = 3\} \twoheadrightarrow \{x = 3 \lor x \leq y\}$
2. $\{x \neq x\} \twoheadrightarrow \{x = 3\}$
3. $\{x \leq y\} \twoheadleftarrow\!\!\twoheadrightarrow \{x < y \lor x = y\}$
4. $\{x = 2[x \mapsto x + 1][x \mapsto 1]\} \twoheadleftarrow\!\!\twoheadrightarrow \{\top\}$

Assertion implication

We say that assertion A *implies* assertion B, notation $A \twoheadrightarrow B$, if, and only if, for any state s, $A(s) \implies B(s)$. Similarly, we say that two assertions are equivalent, notation $A \twoheadleftarrow\!\!> B$, if, and only if, $A(s) \iff B(s)$ for any state s.

1. $\{x = 3\} \twoheadrightarrow \{x = 3 \lor x \leq y\}$
2. $\{x \neq x\} \twoheadrightarrow \{x = 3\}$
3. $\{x \leq y\} \twoheadleftarrow\!\!\twoheadrightarrow \{x < y \lor x = y\}$
4. $\{x = 2[x \mapsto x + 1][x \mapsto 1]\} \twoheadleftarrow\!\!\twoheadrightarrow \{\top\}$

```
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 showed that $\{\top\}\ x ::= 1; ; x ::= x + 1\ \{x = 2\}$.

> Which of the following hold?

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

Weakening and strengthening pre-/post conditions

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

> Which of the following hold?

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

Weakening and strengthening pre-/post conditions

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

> Which of the following hold?

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

Weakening and strengthening pre-/post conditions

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

> Which of the following hold?

1. $\{\mathbf{y} = \mathbf{1}\}\ x ::= 1;;x ::= x+1\ \{x = 2\}$ **Holds.**
2. $\{\mathbf{x} = \mathbf{10}\}\ x ::= 1;;x ::= x+1\ \{x = 2\}$ **Holds.**
3. $\{\top\}\ x ::= 1;;x ::= x+1\ \{x = 2 \wedge \mathbf{y} = \mathbf{1}\}$ **Does NOT hold.**
4. $\{\top\}\ x ::= 1;;x ::= x+1\ \{\top\}$

Weakening and strengthening pre-/post conditions

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

> Which of the following hold?

1. $\{\mathbf{y} = \mathbf{1}\}\ x ::= 1;; x ::= x + 1\ \{x = 2\}$ **Holds.**
2. $\{\mathbf{x} = \mathbf{10}\}\ x ::= 1;; x ::= x + 1\ \{x = 2\}$ **Holds.**
3. $\{\top\}\ x ::= 1;; x ::= x + 1\ \{x = 2 \wedge \mathbf{y} = \mathbf{1}\}$ **Does NOT hold.**
4. $\{\top\}\ x ::= 1;; x ::= x + 1\ \{\top\}$ **Holds.**
5. $\{\top\}\ x ::= 1;; x ::= x + 1\ \{\bot\}$

Weakening and strengthening pre-/post conditions

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

> Which of the following hold?

1. $\{\mathbf{y} = \mathbf{1}\}\ x ::= 1;;x ::= x + 1\ \{x = 2\}$ **Holds.**
2. $\{\mathbf{x} = \mathbf{10}\}\ x ::= 1;;x ::= x + 1\ \{x = 2\}$ **Holds.**
3. $\{\top\}\ x ::= 1;;x ::= x + 1\ \{x = 2 \wedge \mathbf{y} = \mathbf{1}\}$ **Does NOT hold.**
4. $\{\top\}\ x ::= 1;;x ::= x + 1\ \{\top\}$ **Holds.**
5. $\{\top\}\ x ::= 1;;x ::= x + 1\ \{\bot\}$ **Does NOT hold.**

Thus,

Weakening and strengthening pre-/post conditions

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

> Which of the following hold?

1. $\{\mathbf{y = 1}\}\ x ::= 1; ;\ x ::= x + 1\ \{x = 2\}$ **Holds.**
2. $\{\mathbf{x = 10}\}\ x ::= 1; ;\ x ::= x + 1\ \{x = 2\}$ **Holds.**
3. $\{\top\}\ x ::= 1; ;\ x ::= x + 1\ \{x = 2 \land \mathbf{y = 1}\}$ **Does NOT hold.**
4. $\{\top\}\ x ::= 1; ;\ x ::= x + 1\ \{\top\}$ **Holds.**
5. $\{\top\}\ x ::= 1; ;\ x ::= x + 1\ \{\bot\}$ **Does NOT hold.**

Thus,

Theorem (H-cons): If $\{P'\}\ c\ \{Q'\}$, $P \twoheadrightarrow P'$, and $Q' \twoheadrightarrow Q$, then $\{P\}\ c\ \{Q\}$.

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}}
  {{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 $\{P\}$ **IFB** b **THEN** c_1 **ELSE** c_2 **FI** $\{Q\}$.

```
Theorem hoare_cond: forall P Q b c1 c2,
  {{P}} c1 {{Q}} →
  {{P}} c2 {{Q}} →
  {{P}} IFB b THEN c1 ELSE c2 FI {{Q}}.
```

Prove that

$$\frac{\{\top\}\ y ::= 2\ \{x \leq y\} \qquad \{\top\} y ::= x + 1 \{x \leq y\}}{\{\top\}\mathtt{IFB}\ x = 0\ \mathtt{THEN}\ y ::= 2\ \mathtt{ELSE} y ::= x + 1\ \mathtt{FI}\ \{x \leq y\}}\text{H-cond}$$

Conditionals

Proving **ELSE**:

$$\dfrac{\dfrac{\dots}{\{\top\} \twoheadrightarrow \{x \leq y[y \mapsto x+1]\}} \qquad \dfrac{\dots}{\{x \leq y[y \mapsto x+1]\} y ::= x+1 \{x \leq y\}}\text{H-asgn}}{\dfrac{\{\top\} y ::= x+1 \{x \leq y\}}{\{\top\}\texttt{IFB}\ x = 0\ \texttt{THEN}\ y ::= 2\ \texttt{ELSE} y ::= x+1\ \texttt{FI}\ \{x \leq y\}}\text{H-cond}}\text{H-cons-pre}$$

Conditionals

Proving **ELSE**:

$$
\dfrac{
  \dfrac{\dots}{\{\top\} \twoheadrightarrow \{x \leq y[y \mapsto x+1]\}}
  \qquad
  \dfrac{\dots}{\{x \leq y[y \mapsto x+1]\} y ::= x+1 \{x \leq y\}}\text{H-asgn}
}{
  \dfrac{\{\top\} y ::= x+1 \{x \leq y\}}{\{\top\}\texttt{IFB}\ x = 0\ \texttt{THEN}\ y ::= 2\ \texttt{ELSE} y ::= x+1\ \texttt{FI}\ \{x \leq y\}}\text{H-cond}
}\text{H-cons-pre}
$$

Proving **THEN**:

$$
\dfrac{\dfrac{???}{\{\top\}\ y ::= 2\ \{x \leq y\}}}{\{\top\}\texttt{IFB}\ x = 0\ \texttt{THEN}\ y ::= 2\ \texttt{ELSE} y ::= x+1\ \texttt{FI}\ \{x \leq y\}}\text{H-cond}
$$

Conditionals

Proving **ELSE**:

$$\dfrac{\dfrac{\dots}{\{\top\} \twoheadrightarrow \{x \leq y[y \mapsto x+1]\}} \quad \dfrac{\dots}{\{x \leq y[y \mapsto x+1]\} y ::= x+1 \{x \leq y\}}\text{H-asgn}}{\dfrac{\{\top\} y ::= x+1 \{x \leq y\}}{\{\top\}\texttt{IFB}\ x = 0\ \texttt{THEN}\ y ::= 2\ \texttt{ELSE} y ::= x+1\ \texttt{FI}\ \{x \leq y\}}\text{H-cond}}\text{H-cons-pre}$$

Proving **THEN**:

$$\dfrac{\dfrac{???}{\{\top\}\ y ::= 2\ \{x \leq y\}}}{\{\top\}\texttt{IFB}\ x = 0\ \texttt{THEN}\ y ::= 2\ \texttt{ELSE} y ::= x+1\ \texttt{FI}\ \{x \leq y\}}\text{H-cond}$$

> 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 $\{P\}$ `IFB` b `THEN` c_1 `ELSE` c_2 `FI` $\{Q\}$.

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}} (IFB b THEN c1 ELSE c2 FI) {{Q}}.
Proof.
  intros.
```

Example

```
Goal
    {{fun st ⇒ True}}
  IFB X = 0
    THEN Y ::= 2
    ELSE Y ::= X + 1
  FI
    {{fun st ⇒ st X ≤ st Y}}.
```

The Hoare theorem for While

1. $\{P\}\ \texttt{WHILE}\ b\ \texttt{DO}\ c\ \texttt{END}\ \{P\}$

The Hoare theorem for While

1. $\{P\}$ `WHILE` b `DO` c `END` $\{P\}$
2. $\{P\}$ `WHILE` b `DO` c `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. $\{P\}$ `WHILE` b `DO` c `END` $\{P\}$
2. $\{P\}$ `WHILE` b `DO` c `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 $\{P\}$ `WHILE` b `DO` c `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. $\{P\}$ WHILE b DO c END $\{P\}$
2. $\{P\}$ WHILE b DO c 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 $\{P\}$ WHILE b DO c 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 $\{P\}$ WHILE b DO c 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 ⇒ st X ≤ 3}}
  WHILE X ≤ 2
  DO X ::= X + 1 END
    {{fun st ⇒ 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

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

$$\frac{\{P\}\ c_1\ \{Q\} \qquad \{Q\}\ c_2\ \{R\}}{\{P\}\ c_1;;c_2\ \{R\}}\ \text{(H-seq)}$$

$$\frac{P \twoheadrightarrow P' \qquad \{P'\}\ c\ \{Q'\} \qquad Q' \twoheadrightarrow Q}{\{P\}\ c\ \{Q\}}\ \text{(H-cons)}$$

$$\frac{\{P \wedge b\}\ c_1\ \{Q\} \qquad \{P \wedge \neg b\}\ c_2\ \{Q\}}{\{P\}\ \texttt{IFB}\ b\ \texttt{THEN}\ c_1\ \texttt{ELSE}\ c_2\ \texttt{FI}\ \{Q\}}\ \text{(H-if)}$$

$$\frac{\{P \wedge b\}\ c\ \{P\}}{\{P\}\ \texttt{WHILE}\ b\ \texttt{DO}\ c\ \texttt{END}\ \{P \wedge \neg b\}}\ \text{(H-while)}$$

Hoare Logic as an Axiomatic Logic

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

```
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