CS720
Logical Foundations of Computer Science
Lecture 14: Program verification
Tiago Cogumbreiro
Imp.v

Due Thursday October 18, 11:59pm EST
IndProp.v

Due Friday October 19, 11:59pm EST
Equiv.v

Due Thursday October 25, 11:59pm EST
Summary

- Learn how to design a framework to prove properties about programs (We will develop the Floyd-Hoare Logic.)
- Introduce pre and post-conditions on commands
How do we specify an algorithm?
How do we specify an algorithm?

A formal specification describes what a system does (and not how a system does it)
How do we **observe** what an Imp program does?
Specifying Imp programs

The input and the output of an Imp program is a state. Let us call the formalize reasoning about an Imp state as an assertion, notation $\{P\}$, for some proposition $P$ that accesses an implicit state:

Definition Assertion ::= state $\rightarrow$ Prop.

1. $\{x = 3\}$ written as $\text{fun } st \Rightarrow st \; X = 3$
2. $\{x \leq y\}$ written as $\text{fun } st \Rightarrow st \; X \leq st \; Y$
3. $\{x = 3 \lor x \leq y\}$ written as $\text{fun } st \Rightarrow st \; X = 3 \lor st \; X \leq st \; Y$
4. $z \times z \land \neg ((z + 1) \times (z + 1) \leq x)$ written as
   $\text{fun } st \Rightarrow st \; Z \times st \; Z \leq st \; X \lor \neg (((st \; Z)) \times (st \; Z)) \leq st \; X)$
5. What about $\text{fun } st \Rightarrow True$?
6. What about $\text{fun } st \Rightarrow False$?
A Hoare Triple

Combining assertions with commands

A **Hoare triple**, notation \( \{ P \} \ c \ \{ Q \} \), holds if, and only if, from \( P(s) \) and \( c / s \ \\parallel \ s' \) we can obtain \( Q(s') \) for any states \( s \) and \( s' \).

\[
\text{Definition hoare_triple (P:Assertion) (c:com) (Q:Assertion) : Prop :=}
\]
\[
\text{forall st st',}
\]
\[
P \ st \rightarrow (* \text{If } [P \ st] \text{ holds } *)
\]
\[
c / st \ \parallel st' \rightarrow (* \text{And } [c] \text{ runs with an input state } [st] \text{ yielding a state } [st'] \*)
\]
\[
Q \ st'. (* \text{Then } [Q \ st'] \text{ holds } *)
\]
Exercise

Which of these programs are provable?

1. \{\top\} \ x := 5; \ y := 0 \ {x = 5}
2. \{x = 2 \land x = 3\} \ x := 5 \ {x = 0}
3. \{\top\} \ x := x + 1 \ {x = 2}
4. \{\top\} \text{SKIP} \ \{\bot\}
5. \{x = 1\} \text{WHILE} !(x = 0) \text{DO} \ x := x + 1 \text{END} \ {x = 100}
Let us build a theory on Hoare triples over Imp

(That is, define theorems to help us prove results on Hoare triples.)
Skip

Theorem (H-skip): for any proposition $P$ we have that $\{P\} \text{ SKIP } \{P\}$.

Theorem hoare_skip : forall P, 
\{P\} SKIP \{P\}.\n
CS720: Lecture 14  Tiago Cogumbreiro
Sequence

Theorem (H-seq): If \( \{P\} c_1 \{Q\} \) and \( \{Q\} c_2 \{R\} \), then
Sequence

Theorem (H-seq): If $\{P\} c_1 \{Q\}$ and $\{Q\} c_2 \{R\}$, then $\{P\} c_1;c_2 \{R\}$.

Theorem hoare_seq : forall P Q R c1 c2,
  {{P}} c1 {{Q}} →
  {{Q}} c2 {{R}} →
  {{P}} c1;c2 {{R}}.
We have seen how to derive theorems for some commands,

Let us derive a theorem over the assignment
Assignment

How do we derive a general-enough theorem over the assignment?

**Idea:** try to prove \( \text{False} \) and simplify the hypothesis.

**Goal** for all \( P \),

\[
\{\{ \text{fun } st \Rightarrow P \, \text{st} \} \} X ::= a \{\{ \text{fun } st \Rightarrow P \, \text{st} \lor \text{False} \}\}.
\]

How do we mention pre-updates?
Reasoning about pre-update

\[
\text{Goal}\ \forall P\ m\ a,\\\{\{\text{fun st} \Rightarrow P\ st \land st\ X = m\}\}\\\X ::= a\\\{\{\text{fun st} \Rightarrow P\ st\}\}.
\]
Reasoning about pre-update

\textbf{Goal} \forall P \ m \ a, \\
\{\{ \text{fun} \ st \Rightarrow P \ st \ \land \ st \ X = m \} \} \\
X ::= a \\
\{\{ \text{fun} \ st \Rightarrow P \ st \} \}.

we are stuck here

\textbf{H:} \ st \ X = m \\
\textbf{H0 :} P \ st \\
\hline 
\begin{align*}
P \ (st \ & \land \ \{X \rightarrow \ \text{aeval} \ st \ a\})
\end{align*}

What happens if we change our post-condition?
Let us change the post-condition to understand how it affects our goal

\[
\text{Goal for all } P \ a \ m,
\{
\{ \text{fun } st \Rightarrow P \ st \land st \ X = m \}\}
X ::= a
\{
\{ \text{fun } st \Rightarrow P (st \& \{ X \rightarrow 3 \}) \}\}.
\]

Updating the store of the post-condition *shadows* the update to \( a \)

H: \( st \ X = m \)
H0: \( P \ st \)
\[\text{------------------------------}(1/1)\]
\( P (st \& \{ X \rightarrow \text{aeval st a}; X \rightarrow 3\}) \)
Second try

Let us change the post-condition to understand how it affects our goal

\[
\text{Goal}\: \forall\: P\ a\ m,\\
\{\{\text{fun}\ st \Rightarrow P\ st /\ st\ X = m\}\}\\\
X ::= a\\
\{\{\text{fun}\ st \Rightarrow P\ (st &\{X \rightarrow 3\})\}\}\.
\]

Updating the store of the post-condition *shadows* the update to *a*

<table>
<thead>
<tr>
<th>H: [st\ X = m]</th>
<th>H0: [P\ st]</th>
</tr>
</thead>
<tbody>
<tr>
<td>\hline</td>
<td>\hline</td>
</tr>
<tr>
<td>\hline</td>
<td>\hline</td>
</tr>
</tbody>
</table>

What if we "cancel out" the update?
Reasoning about the post-update

Goal \( \forall P \ a \ m, \)
\[
\{\{ \text{fun } st \Rightarrow P \ st \land st \ X = m \}\}
\]
\( X ::= a \)
\[
\{\{ \text{fun } st \Rightarrow P \ (st \land \{ X \rightarrow m \}) \}\}\]
We are still not there yet. How do we derive the post-value?
Reasoning about the post-update

Goal

\[
\forall P, a, m. \\
\{\{ \text{fun} \ st \Rightarrow P \ st \land st \ X = m \}\} \\
\ X ::= a \\
\{\{ \text{fun} \ st \Rightarrow P \ (st \land \{ X \rightarrow m \}) \}\}.
\]

We are still not there yet. How do we derive the post-value?

Theorem

\[
\text{hoare\_asgn\_fwd} : \\
\forall m, a, P. \\
\{\{ \text{fun} \ st \Rightarrow P \ st \land st \ X = m \}\} \\
\ X ::= a \\
\{\{ \text{fun} \ st \Rightarrow P \ (st \land \{ X \rightarrow m \}) \land st \ X = \text{aeval} \ (st \land \{ X \rightarrow m \}) a \}\}.
\]

This would be a very difficult theorem to apply. Can we do better?
Rephrasing the assignment rule

Recall that

\[
\text{Goal for all } P \text{ m a,} \\
\{\{ \text{fun st } \Rightarrow P \text{ st } \}\} X ::= a \{\{ \text{fun st } \Rightarrow P \text{ st } \}\}.
\]

lead us here

\[
H0 : P \text{ st} \\
\text{--------------------------(1/1)} \\
P (\text{st } \& \{X \rightarrow \text{aeval st } a\})
\]

What if we update the store in the pre-condition?
Rephrasing the pre-condition

**Goal**
\[ \forall P \, m \, a, \{\{ \text{fun} \, st \Rightarrow P (st \land \{X \rightarrow 3\}) \}\} \, X \leftarrow a \{\{ \text{fun} \, st \Rightarrow P \, st \}\}. \]

leads us here

\[ H_0 : P (st \land \{X \rightarrow 3\}) \]

\[ \text{-----------------------------------------------(1/1)} \]
\[ P (st \land \{X \rightarrow \text{aeval} \, st \, a\}) \]

Why not just set the pre-condition to \( P (st \land \{X \rightarrow \text{aeval} \, st \, a\}) \)?
Backward style assignment rule

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

Theorem hoare_asgn: $\forall a \ P, \{\{ \text{fun} \ st \Rightarrow P (st \ & \ \{X \mapsto \text{aeval st a}\})\}\}} \ X ::= a \\ \{\{ \text{fun} \ st \Rightarrow P \ st \}\}$. 
Exercise

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

Goal \( \{\text{(fun st : state } \Rightarrow \text{ st } X = 2) \ [X \mapsto X + 1] \ [X \mapsto 1] \} \)

\[ X ::= 1 ; ; X ::= X + 1 \]

\( \{\text{fun st } \Rightarrow \text{ st } X = 2 \} \).
Exercise

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

\[
\text{Goal} \ \{\ (\text{fun} \ st : \text{state} \Rightarrow \text{st} \ x = 2) \ [X \mapsto X + 1] \ [X \mapsto 1] \ \}\ \\
X ::= 1 ; ; X ::= X + 1 \\
\{\ \text{fun} \ st \Rightarrow \text{st} \ x = 2 \ \}.
\]

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

\[
\text{Goal} \ \{\ \text{fun} \ st \Rightarrow \text{True} \ \}\ \\
X ::= 1 ; ; X ::= X + 1 \\
\{\ \text{fun} \ st \Rightarrow \text{st} \ x = 2 \ \}.
\]
Exercise

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

\[ \text{Goal } \{\text{ (fun st : state } \Rightarrow \text{ st } X = 2) [X |\mapsto X + 1] [X |\mapsto 1] \} \]
\[ X ::= 1;; X ::= X + 1 \{\text{ fun st } \Rightarrow \text{ st } X = 2 \}. \]

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

\[ \text{Goal } \{\text{ fun st } \Rightarrow \text{ True } \} \ x ::= 1;; X ::= X + 1 \{\text{ fun st } \Rightarrow \text{ st } X = 2 \}. \]

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.
Summary

Here are theorems we've proved today:

\[
\{P\} \text{ SKIP } \{P\} \quad \text{(H-skip)}
\]

\[
\begin{array}{c}
\{P\} \quad c_1 \quad \{Q\} \quad c_2 \quad \{R\} \\
\hline
\{P\} \quad c_1 \quad ; \quad c_2 \quad \{R\}
\end{array}
\quad \text{(H-seq)}
\]

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

- Learn how to design a framework to prove properties about programs (We will develop the Floyd-Hoare Logic.)
- Introduce pre and post-conditions on commands