Introduction to Deductive Program Verification

Işıl Dillig

Hoare Logic

- Hoare logic forms the basis of all deductive verification techniques
- Named after Tony Hoare: inventor of quick sort, father of formal verification, 1980 Turing award winner
- Logic is also known as Floyd-Hoare logic: some ideas introduced by Robert Floyd in 1967 paper “Assigning Meaning to Programs”

Simple Imperative Programming Language

- To illustrate Hoare logic, we’ll consider a small imperative programming language IMP
- In IMP, we distinguish three program constructs: expressions, conditionals, and statements
- Expression \( E := Z \mid V \mid e_1 + e_2 \mid e_1 \times e_2 \)
- Conditional \( C := true \mid false \mid e_1 = e_2 \mid e_1 \leq e_2 \)
- Statement \( S := V := E \) (Assignment)
- \( S_1 ; S_2 \) (Composition)
- if \( C \) then \( S_1 \) else \( S_2 \) (If)
- while \( C \) do \( S \) (While)

Partial Correctness Specification

- In Hoare logic, we specify partial correctness of programs using Hoare triples: \( \{ P \} S \{ Q \} \)
- Here, \( S \) is a statement in programming language IMP
- \( P \) and \( Q \) are SMT formulas
- \( P \) is called precondition and \( Q \) is called post-condition

Meaning of Hoare Triples

- Meaning of Hoare triple \( \{ P \} S \{ Q \} \):
  - If \( S \) is executed in state satisfying \( P \)
  - and if execution of \( S \) terminates
  - then the program state after \( S \) terminates satisfies \( Q \)
- Is \( \{ x = 0 \} x := x + 1 \{ x = 1 \} \) valid Hoare triple? yes
- What about \( \{ x = 0 \land y = 1 \} x := x + 1 \{ x = 1 \land y = 2 \} \)? no
- What about \( \{ x = 0 \} x := x + 1 \{ x = 1 \lor y = 2 \} \)? yes
- What about \( \{ x = 0 \} \) while true do \( x := 0 \{ x = 1 \} \)? yes
Partial vs. Total Correctness

- The specification \( \{P\}S\{Q\} \) called partial correctness spec. b/c doesn’t require \( S \) to terminate
- There is also a stronger requirement called total correctness
- Total correctness specification written \( \{P\}S\{Q\} \)
- Meaning of \( \{P\}S\{Q\} \):
  - If \( S \) is executed in state satisfying \( P \)
  - then the execution of \( S \) terminates
  - and program state after \( S \) terminates satisfies \( Q \)
- Is \( \{x = 0\} \) while true do \( x := 0 \) \( \{x = 1\} \) valid? no

Example Specifications

- What does \( \{\text{true}\}S\{Q\} \) say? If \( S \) terminates, then \( Q \) holds
- What about \( \{P\}S\{\text{true}\} \)? holds for any \( P \) and any \( S \)
- What about \( \{P\}S\{\text{true}\} \)? If \( P \) holds, then \( S \) terminates
- When does \( \{\text{true}\}S\{\text{false}\} \) hold? If \( S \) does not terminate
- When does \( \{\text{false}\}S\{Q\} \) hold? Always
- We’ll only focus on only partial correctness (safety)
- Total correctness = Partial correctness + termination

More Examples

Valid or invalid?

- \( \{i = 0\} \) while \( i < n \) do \( i++\); \( \{i = n\} \) no
- \( \{i = 0\} \) while \( i < n \) do \( i++\); \( \{i \geq n\} \) yes
- \( \{i = 0 \land j = 0\} \) while \( i < n \) do \( i++\); \( j \leftarrow i \); \( \{2i = n(n + 1)\} \) no
- How can we strengthen the precondition so it’s valid? add \( n \geq 0 \)

Inference Rules

- Proof rules in Hoare logic are written as inference rules:

\[
\frac{\vdash \{P_1\}S_1\{Q_1\} \cdots \vdash \{P_n\}S_n\{Q_n\}}{\vdash \{P\}S\{Q\}}
\]

- Says if Hoare triples \( \{P_1\}S_1\{Q_1\}, \ldots, \{P_n\}S_n\{Q_n\} \) are provable in our proof system, then \( \{P\}S\{Q\} \) is also provable.
- Not all rules have hypotheses: these correspond to bases cases in the proof
- Rules with hypotheses correspond to inductive cases in proof
- One inference rule for every statement in the IMP language

Proving Partial Correctness

- Key problem: How to prove valid Hoare triples?
- If a Hoare triple is valid, written \( \models \{P\}S\{Q\} \), we want a proof system to prove its validity
- Use notation \( \vdash \{P\}S\{Q\} \) to indicate that we can prove validity of Hoare triple
- Hoare also gave a sound and (relatively-) complete proof system that allows semi-mechanizing correctness proofs
- Soundness: If \( \vdash \{P\}S\{Q\} \), then \( \models \{P\}S\{Q\} \)
- Completeness: If \( \models \{P\}S\{Q\} \), then \( \vdash \{P\}S\{Q\} \)

Understanding Proof Rule for Assignment

- Consider the assignment \( x := y \) and post-condition \( x > 2 \)
- What do we need before the assignment so that \( x > 2 \) holds afterwards? \( y > 2 \)
- Consider \( i := i + 1 \) and post-condition \( i > 10 \)
- What do we need to know before the assignment so that \( i > 10 \) holds afterwards? \( i > 9 \)
Proof Rule for Assignment

\[ \vdash \{Q[x/E]\} x := E \{Q\} \]

- To prove \(Q\) holds after assignment \(x := E\), sufficient to show that \(Q\) with \(E\) substituted for \(x\) holds before the assignment.
- Using this rule, which of these are provable?
  - \(\{y = 4\} x := 4 \{y = x\}\) yes
  - \(\{x + 1 = n\} x := x + 1 \{x = n\}\) yes
  - \(\{y = x\} y := 2 \{y = x\}\) no
  - \(\{z = 3\} y := x \{z = 3\}\) yes

Motivation for Precondition Strengthening

- Is the Hoare triple \(\{z = 2\} y := x \{y = x\}\) valid? yes
- Is this Hoare triple provable using our assignment rule? no
- Instantiating the assignment rule, we get:
  \(\{y = x[x/y] x = x \text{true}\} y = x \{y = x\}\)
- But intuitively, if we can prove \(y = x\) w/o any assumptions, we should also be able to prove it if we do make assumptions!

Proof Rule for Precondition Strengthening

\[ \vdash \{P'\} S \{Q\} \quad P \Rightarrow P' \]

\[ \vdash \{P\} S \{Q\} \]

- Recall: \(P \Rightarrow P'\) means the formula \(P \rightarrow P'\) is valid
- Hence, need to use SMT solver every time we use precondition strengthening!

Example

- Using this rule and rule for assignment, we can now prove \(\{z = 2\} y := x \{y = x\}\)
- Proof:
  \[ \vdash \{y = x[x/y]\} y = x \{y = x\} \]
  \[ \vdash \{\text{true}\} y := x \{y = x\} \]
  \[ \vdash \{z = 2\} y := x \{y = x\} \]

Proof Rule for Post-Condition Weakening

\[ \vdash \{P\} S \{Q'\} \quad Q' \Rightarrow Q \]

\[ \vdash \{P\} S \{Q\} \]

- We also need a dual rule for post-conditions called post-condition weakening:
- If we can prove post-condition \(Q'\), we can always relax it to something weaker
- Again, need to use SMT solver when applying post-condition weakening
**Proof Rule for If Statements**

- Suppose we know \( P \) holds before if statement and want to show \( Q \) holds afterwards.
- At beginning of then branch, what facts do we know? \( P \land C \)
- Thus, in the then branch, we want to show \( \{ P \land C \} S_1 \{ Q \} \)
- At beginning of else branch, what facts do we know? \( P \land \neg C \)
- What do we need to show in else branch? \( \{ P \land \neg C \} S_2 \{ Q \} \)

**Exercise, cont**

- Yes, this rule can be derived from existing rules.
- From premises, we know (1) \( \{ P \land C \} S_1 \{ Q \} \) and (2) \( \{ P \land \neg C \} S_2 \{ Q \} \)
- Let \( Q' \) be the weakest precondition we need for \( Q \) to hold after executing \( S_1 \), i.e., (3) \( \{ Q' \} S_1 \{ Q \} \)
- Using premises, this means \( \{ P \land C \} S_1 \{ Q' \} \) and \( \{ P \land \neg C \} S_2 \{ Q' \} \)
- From these and existing if rule, we can derive:
  \( \{ P \} \{ \text{if } C \text{ then } S_1 \text{ else } S_2 \} \{ Q' \} \)
- Conclusion follows from these and (3) using Seq

**Exercise**

- Your friend suggests the following proof rule:
  \[
  \begin{align*}
  &\{ P \land C \} S_1; S_3 \{ Q \} \\
  &\{ P \land \neg C \} S_2; S_3 \{ Q \}
  \end{align*}
  \]
  \( \{ P \} \{ \text{if } C \text{ then } S_1 \text{ else } S_2 \} ; S_3 \{ Q \} \)
- Is this proof rule correct? If so, prove your answer. Otherwise, give a counterexample.
Proof Rule for While and Loop Invariants

- Last proof rule of Hoare logic is that for while loops.
- But to understand proof rule for while, we first need concept of a loop invariant.
- A loop invariant $I$ has following properties:
  1. $I$ holds initially before the loop.
  2. $I$ holds after each iteration of the loop.

Proof Rule for While

- Consider the statement while $C$ do $S$.
- Suppose $I$ is a loop invariant for this loop. What is guarantee of proof rule for while with loop terminates? $I \land \lnot C$.
- Putting all this together, proof rule for while is:
  $$\vdash \{P \land C\} S(P) \quad \vdash \{P\} \text{while } C \text{ do } S(P \land \lnot C).$$
- This rule simply says "If $P$ is a loop invariant, then $P \land \lnot C$ must hold after loop terminates".
- Based on this rule, why is $P$ a loop invariant?
- Because $P$ holds initially and is preserved after each iteration.

Example, cont.

- Ok, we've shown $x \leq n$ is loop invariant, now let's instantiate proof rule for while with this loop invariant:
  $$\vdash \{x \leq n \land x < n\} S'(x \leq n)$$
- Recall: We wanted to prove the Hoare triple $\{x \leq n\} S(x \geq n)$.
- In addition to proof rule for while, what other rule do we need? postcondition weakening.

Proof Rule for While and Loop Invariants

- Last proof rule of Hoare logic is that for while loops.
- But to understand proof rule for while, we first need concept of a loop invariant.
- A loop invariant $I$ has following properties:
  1. $I$ holds initially before the loop.
  2. $I$ holds after each iteration of the loop.

Proof Rule for While

- Consider the statement while $C$ do $S$.
- Suppose $I$ is a loop invariant for this loop. What is guarantee of proof rule for while with loop terminates? $I \land \lnot C$.
- Putting all this together, proof rule for while is:
  $$\vdash \{P \land C\} S(P) \quad \vdash \{P\} \text{while } C \text{ do } S(P \land \lnot C).$$
- This rule simply says "If $P$ is a loop invariant, then $P \land \lnot C$ must hold after loop terminates".
- Based on this rule, why is $P$ a loop invariant?
- Because $P$ holds initially and is preserved after each iteration.

Example

- Consider the statement $S = \text{while } x < n \text{ do } x = x + 1$
- Let's prove validity of $\{x \leq n\} S(x \geq n)$.
- What is appropriate loop invariant? $x \leq n$
- First, let's prove $x \leq n$ is loop invariant. What do we need to show? $\{x \leq n \land x < n\} x = x + 1 \{x \leq n\}$
- What proof rules do we need to use to show this? assignment, precondition strengthening
  $$\vdash \{x \leq n \land x < n\} x = x + 1 \{x \leq n\} x = x + 1 \{x \leq n\}$$

Example, cont.

- Consider the following code
  $$i := 0; j := 0; n := 10; \text{while } i < n \text{ do } i := i + 1; j := i + j$$
- Which of the following are loop invariants?
  1. $i \leq n$ yes
  2. $i < n$ no
  3. $j \geq 0$ yes
- Suppose $I$ is a loop invariant. Does $I$ also hold after loop terminates?
- Yes because, by definition, $I$ holds after every loop iteration, including after the last one.
Invariant vs. Inductive Invariant

- Suppose $I$ is a loop invariant for while $C$ do $S$.
- Does it always satisfy $\{I \land C\}S\{I\}$?
- Counterexample: Consider $I = j \geq 1$ and the code:
  \[
i := 1; j := 1; \text{while } i < n \text{ do } \{j := j + i; i := i + 1\}\]
- But strengthened invariant $j \geq 1 \land i \geq 1$ does satisfy it
- Such invariants are called inductive invariants, and they are the only invariants that we can prove
- Key challenge in verification is finding inductive loop invariants

Exercise

Find inductive loop invariant to prove the following Hoare triple:
\[
\{i = 0 \land j = 0 \land n = 5\} \quad \text{while } i < n \text{ do } i := i + 1; j := j + 1 \quad \{j = 15\}
\]
- Inductive loop invariant $I$:
  \[
  2j = i(i + 1) \land i \leq n \land n = 5
  \]
- Weakest precondition $P$ w.r.t loop body:
  \[
  2j = i(i + 1) \land i + 1 \leq n \land n = 5
  \]
- Since $I \land C \Rightarrow P$, $I$ is inductive.

Another Exercise

- Suppose we add a for loop construct to IMP:
  \[
  \text{for } v := e_1 \text{ until } e_2 \text{ do } S
  \]
- Initializes $v$ to $e_1$, increments $v$ by 1 in each iteration and terminates when $v > e_2$
- Write a proof rule for this for loop construct
- We can de-sugar into while loop:
  \[
  v := e_1; \text{while } v \leq e_2 \text{ do } \{S; v := v + 1\}
  \]
- Putting all this together, we get the following proof rule:
  \[
  \{P\} v := e_1 \{I\} \\
  \{I \land v \leq e_2\} S; v := v + 1 \{I\} \\
  \{P\} \text{ for } v := e_1 \text{ until } e_2 \text{ do } S \{I \land v > e_2\}
  \]