Deductive Program Verification using Hoare Logic

Andrei Arusoaie

What is Deductive Program Verification?

- Deductive program verification is a method to prove the correctness of programs using formal logic
- We need to specify what a program is supposed to do using logical formulas
- Examples: safety (no crashes), no arithmetic over/underflows, behavioral properties
- The program is verified by *proving* that these specifications (logical formulas) hold throughout the execution of the program

Why Deductive Program Verification is Useful?

- **Correctness**: Helps ensure that a program behaves as intended
- **Bug detection**: Formal methods can detect edge cases and bugs missed by testing
- Especially important for safety-critical systems (e.g., aviation, medical devices)
- **Drawback**: Requires higher expertise

(Floyd-)Hoare Logic

- Robert Floyd, 1967:
  `https://people.eecs.berkeley.edu/~necula/Papers/FloydMeaning.pdf`

(Floyd-)Hoare Logic

- Robert Floyd, 1967: `https://people.eecs.berkeley.edu/~necula/Papers/FloydMeaning.pdf`
- Set the foundation of axiomatic semantics of classical programs

(Floyd-)Hoare Logic

- Robert Floyd, 1967: `https://people.eecs.berkeley.edu/~necula/Papers/FloydMeaning.pdf`
- Set the foundation of axiomatic semantics of classical programs
- C. A. R. Hoare, 1969: `https://dl.acm.org/doi/pdf/10.1145/363235.363259`
- Hoare is the inventor of quick sort, father of formal verification, Turing award winner 1980

Imperative Programming Language (IMP)

Arithmetic Expressions:

$$\mathsf{AExp} ::= \mathsf{Var} \mid \mathsf{Int} \mid \mathsf{AExp} + \mathsf{AExp} \mid \mathsf{AExp} \;/\; \mathsf{AExp}$$

Boolean Expressions:

$$\begin{aligned} \mathsf{BExp} ::= {} & \mathit{true} \mid \mathit{false} \mid \mathsf{AExp} < \mathsf{AExp} \\ & \mid \mathit{not}\ \mathsf{BExp} \mid \mathsf{BExp}\ \mathit{and}\ \mathsf{BExp} \end{aligned}$$

Statements:

$$\begin{aligned} \mathsf{Stmt} ::= {} & \mathsf{Var} := \mathsf{AExp} \\ & \mid \mathit{if}\ \mathsf{BExp}\ \mathit{then}\ \mathsf{Stmt}\ \mathit{else}\ \mathsf{Stmt} \\ & \mid \mathit{while}\ \mathsf{BExp}\ \mathit{do}\ \mathsf{Stmt}\ \mathit{end} \\ & \mid \mathsf{Stmt}\ ;\ \mathsf{Stmt} \mid \mathit{skip} \end{aligned}$$

Deductive Program Verification Workflow

- **Step 1:** Write specifications for the program
- **Step 2:** Use Hoare logic to write the specs (e.g., *preconditions* and *postconditions*)
- **Step 3:** Generate verification conditions that prove the program's correctness
- **Step 4:** Use theorem provers to check if verification conditions hold

The sumPgm Program

Program:

```
sum := 0;
i := 1;
while (i < n + 1) do
    sum := sum + i;
    i := i + 1
end
```

Input constraint: $n \geq 0$

The sumPgm Program

Program:

```
sum := 0;
i := 1;
while (i < n + 1) do
    sum := sum + i;
    i := i + 1
end
```

Input constraint: $n \geq 0 \leftarrow$ called *precondition*

The `sumPgm` Program

Program:

```
sum := 0;
i := 1;
while (i < n + 1) do
    sum := sum + i;
    i := i + 1
end
```

Input constraint: $n \geq 0 \leftarrow$ called *precondition*

Desired output property: $sum = \frac{n(n+1)}{2}$

The sumPgm Program

Program:

```
sum := 0;
i := 1;
while (i < n + 1) do
    sum := sum + i;
    i := i + 1
end
```

Input constraint: $n \geq 0 \leftarrow$ called *precondition*

Desired output property: $sum = \frac{n(n+1)}{2} \leftarrow$ called *postcondition*

Hoare Triples

- We can write specifications using Hoare triples:

$$\{P\}\ S\ \{Q\}$$

Hoare Triples

- We can write specifications using Hoare triples:

$$\{P\}\ S\ \{Q\}$$

- P is the precondition
- S is the program
- Q is the postcondition

Hoare Triples

- We can write specifications using Hoare triples:

$$\{P\}\ S\ \{Q\}$$

- P is the precondition
- S is the program
- Q is the postcondition

Hoare Triple for our program:

$$\{n \geq 0\}\ \texttt{sumPgm}\ \{sum = \frac{n(n+1)}{2}\}$$

Hoare Triples: Semantics

- How to establish the validity of a Hoare triple $\{P\}\ S\ \{Q\}$?

Hoare Triples: Semantics

- How to establish the validity of a Hoare triple $\{P\}\ S\ \{Q\}$?
- **Partial correctness**: If S is executed in a state satisfying P and the execution of S terminates, then the resulting program state satisfies Q
  Valid (in the sense of partial correctness) triples are denoted:

$$\models \{P\}\ S\ \{Q\}$$

Hoare Triples: Semantics

- How to establish the validity of a Hoare triple $\{P\}\ S\ \{Q\}$?
- **Partial correctness**: If S is executed in a state satisfying P and the execution of S terminates, then the resulting program state satisfies Q
  Valid (in the sense of partial correctness) triples are denoted:

$$\models \{P\}\ S\ \{Q\}$$

- **Total correctness**: If S is executed in a state satisfying P, then the execution of S terminates and the resulting program state satisfies Q (usually denoted by $[P]\ S\ [Q]$)
  Valid (in the sense of total correctness) triples are denoted:

$$\models [P]\ S\ [Q]$$

Hoare Triples: Semantics

- How to establish the validity of a Hoare triple $\{P\}\ S\ \{Q\}$?
- **Partial correctness**: If S is executed in a state satisfying P and the execution of S terminates, then the resulting program state satisfies Q
  Valid (in the sense of partial correctness) triples are denoted:
  $$\models \{P\}\ S\ \{Q\}$$
- **Total correctness**: If S is executed in a state satisfying P, then the execution of S terminates and the resulting program state satisfies Q (usually denoted by $[P]\ S\ [Q]$)
  Valid (in the sense of total correctness) triples are denoted:
  $$\models [P]\ S\ [Q]$$
- We only discuss partial correctness (safety) in this material
  Total correctness = Partial correctness + termination

A Few Interesting Examples

- $\{\texttt{true}\}\ S\ \{Q\}$

A Few Interesting Examples

- $\{\texttt{true}\}\ S\ \{Q\}$
- $\{P\}\ S\ \{\texttt{true}\}$

A Few Interesting Examples

- $\{\texttt{true}\}\ S\ \{Q\}$
- $\{P\}\ S\ \{\texttt{true}\}$
- $[P]\ S\ [\texttt{true}]$

A Few Interesting Examples

- $\{\texttt{true}\}\ S\ \{Q\}$
- $\{P\}\ S\ \{\texttt{true}\}$
- $[P]\ S\ [\texttt{true}]$
- $\{\texttt{true}\}\ S\ \{\texttt{false}\}$

A Few Interesting Examples

- $\{\texttt{true}\}\ S\ \{Q\}$
- $\{P\}\ S\ \{\texttt{true}\}$
- $[P]\ S\ [\texttt{true}]$
- $\{\texttt{true}\}\ S\ \{\texttt{false}\}$
- $\{\texttt{false}\}\ S\ \{Q\}$

Which of the Following Triples are Valid?

- $\{i = 0\}$ `while` $i < n$ `do` $i := i + 1$ `end` $\{i = n\}$

Which of the Following Triples are Valid?

- $\{i = 0\}$ `while` $i < n$ `do` $i := i + 1$ `end` $\{i = n\}$
- $\{i = 0\}$ `while` $i < n$ `do` $i := i + 1$ `end` $\{i \geq n\}$

Which of the Following Triples are Valid?

- $\{i = 0\}$ `while` $i < n$ `do` $i := i + 1$ `end` $\{i = n\}$
- $\{i = 0\}$ `while` $i < n$ `do` $i := i + 1$ `end` $\{i \geq n\}$
- $\{i = 0 \wedge s = 0\}$

  ```
  while i < n do
      i := i + 1;
      s := s + i
  end
  ```

  $\{2 \cdot s = n(n + 1)\}$

Proofs

- Challenge: how do we prove the validity of triples?

Proofs

- Challenge: how do we prove the validity of triples? We need a proof system!

Proofs

- Challenge: how do we prove the validity of triples? We need a proof system!
- We want a proof system to help with proving $\models \{P\}\ S\ \{Q\}$

Proofs

- Challenge: how do we prove the validity of triples? We need a proof system!
- We want a proof system to help with proving $\models \{P\}\ S\ \{Q\}$
- We use $\vdash \{P\}\ S\ \{Q\}$ to denote a *sequent*

Proofs

- Challenge: how do we prove the validity of triples? We need a proof system!
- We want a proof system to help with proving $\models \{P\}\ S\ \{Q\}$
- We use $\vdash \{P\}\ S\ \{Q\}$ to denote a *sequent*
- $\vdash$ denotes syntactical calculus
- $\models$ indicates semantic validity

Proof System Properties

- The proof system of Hoare Logic includes several proof rules which depend on the language constructs
- Hoare proved that his proof system is correct and (relatively-)complete
- Hoare also gave a sound and (relatively-)complete proof system that allows semi-mechanizing correctness proofs
- **Soundness:** If $\vdash \{P\}\ S\ \{Q\}$, then $\models \{P\}\ S\ \{Q\}$
- Unfortunately, completeness does **not** hold:
  If $\models \{P\}\ S\ \{Q\}$, then $\vdash \{P\}\ S\ \{Q\}$

Inference Rules

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

$$\frac{\vdash \{P_1\}\ S_1\ \{Q_1\} \quad \dots \quad \vdash \{P_n\}\ S_n\ \{Q_n\}}{\vdash \{P\}\ S\ \{Q\}}$$

- The sequents above the horizontal line are premises, while the sequent under the line is the conclusion
- The above rule says: if the sequents

$$\vdash \{P_1\}\ S_1\ \{Q_1\}, \dots, \vdash \{P_n\}\ S_n\ \{Q_n\}$$

  are provable in our proof system, then

$$\vdash \{P\}\ S\ \{Q\}$$

  is also provable
- Rules with no hypotheses are called axioms

Understanding Proof Rule for Assignment

- There is one inference rule for every statement in the IMP language
- Assignments change the value of a variable in the state
- When `x := e` is executed, what must be true before the assignment for a postcondition $\{Q\}$ to hold?
- Remark: $\{Q\}$ holds *after* the assignment!

Understanding Proof Rule for Assignment

- There is one inference rule for every statement in the IMP language
- Assignments change the value of a variable in the state
- When `x := e` is executed, what must be true before the assignment for a postcondition $\{Q\}$ to hold?
- Remark: $\{Q\}$ holds *after* the assignment!
- So, e must be substituted for `x` in $\{Q\}$ before the assignment

Proof Rule for Assignment

$$\frac{\cdot}{\vdash \{Q[e/x]\}\ \texttt{x := e}\ \{Q\}}$$

- This means: if $\{Q[e/x]\}$ holds before the assignment, then after the assignment `x := e`, the postcondition $\{Q\}$ will hold
- Recall that $Q[e/x]$ is the notation for substitution: the postcondition Q with every occurrence of `x` replaced by the expression `e`

Example 1

Is this sequent valid: $\vdash \{\texttt{true}\}\ i := 2\ \{i = 2\}$?

Example 1

Is this sequent valid: $\vdash \{\texttt{true}\}\ i := 2\ \{i = 2\}$?

We use the assignment rule:

$$\frac{.}{\vdash \{Q[e/x]\}\ \texttt{x := e}\ \{Q\}}$$

Example 1

Is this sequent valid: $\vdash \{\texttt{true}\}\ i := 2\ \{i = 2\}$?

We use the assignment rule:

$$\frac{\cdot}{\vdash \{\, Q\,[\,e\,/\,x\,]\}\ \texttt{x}\ \ \texttt{:=}\ \ \texttt{e}\ \{\, Q\,\}}$$

to obtain:

$$\frac{\cdot}{\vdash \{\, i = 2\,[\,2\,/\,i\,]\}\ \texttt{i}\ \ \texttt{:=}\ \ 2\ \{\, i = 2\,\}}$$

Example 1

Is this sequent valid: $\vdash \{\texttt{true}\}\ i := 2\ \{i = 2\}$?
We use the assignment rule:

$$\frac{.}{\vdash \{Q[e/x]\}\ \texttt{x} := \texttt{e}\ \{Q\}}$$

to obtain:

$$\frac{.}{\vdash \{i = 2[2/i]\}\ \texttt{i} := \texttt{2}\ \{i = 2\}}$$

We compute the substitution, and we get:

$$\frac{.}{\vdash \{2 = 2\}\ \texttt{i} := \texttt{2}\ \{i = 2\}}$$

Example 1

Is this sequent valid: $\vdash \{\texttt{true}\}\ i := 2\ \{i = 2\}$?
We use the assignment rule:

$$\frac{\cdot}{\vdash \{Q[e/x]\}\ \texttt{x}\ \texttt{:=}\ \texttt{e}\ \{Q\}}$$

to obtain:

$$\frac{\cdot}{\vdash \{i = 2[2/i]\}\ \texttt{i}\ \texttt{:=}\ \texttt{2}\ \{i = 2\}}$$

We compute the substitution, and we get:

$$\frac{\cdot}{\vdash \{2 = 2\}\ \texttt{i}\ \texttt{:=}\ \texttt{2}\ \{i = 2\}}$$

This results in:

$$\frac{\cdot}{\vdash \{\texttt{true}\}\ \texttt{i}\ \texttt{:=}\ \texttt{2}\ \{i = 2\}}$$

Example 2

Is this sequent valid: $\vdash \{i = k\}\ i := i + 1\ \{i = k + 1\}$?

Example 2

Is this sequent valid: $\vdash \{i = k\}\ i := i + 1\ \{i = k + 1\}$?
We use the assignment rule:

$$\frac{.}{\vdash \{Q[e/x]\}\ \mathtt{x}\ := \mathtt{e}\ \{Q\}}$$

Example 2

Is this sequent valid: $\vdash \{i = k\}\ i := i + 1\ \{i = k + 1\}$?
We use the assignment rule:

$$\frac{.}{\vdash \{Q[e/x]\}\ \mathtt{x} \ \mathtt{:=}\ \mathtt{e}\ \{Q\}}$$

to obtain:

$$\frac{.}{\vdash \{i = k + 1[i + 1/i]\}\ \mathtt{i}\ \mathtt{:=}\ \mathtt{i\ +\ 1}\ \{i = k + 1\}}$$

Example 2

Is this sequent valid: $\vdash \{i = k\}\ i := i + 1\ \{i = k + 1\}$?
We use the assignment rule:

$$\frac{.}{\vdash \{\, Q\,[\, e\,/\,x\,]\,\}\ \mathtt{x}\ \ \mathtt{:=}\ \ \mathtt{e}\ \{\, Q\,\}}$$

to obtain:

$$\frac{.}{\vdash \{\, i = k + 1\,[\, i + 1\,/\,i\,]\,\}\ \mathtt{i}\ \ \mathtt{:=}\ \ \mathtt{i\ +\ 1}\ \{\, i = k + 1\,\}}$$

We compute the substitution, and we get:

$$\frac{.}{\vdash \{i + 1 = k + 1\}\ \mathtt{i}\ \ \mathtt{:=}\ \ \mathtt{i\ +\ 1}\ \{i = k + 1\}}$$

Example 2

Is this sequent valid: $\vdash \{i = k\}\ i := i + 1\ \{i = k + 1\}$?
We use the assignment rule:

$$\frac{.}{\vdash \{\boxed{Q}[\boxed{e}/\boxed{x}]\}\ \boxed{\texttt{x}}\ \texttt{:=}\ \boxed{\texttt{e}}\ \{\boxed{Q}\}}$$

to obtain:

$$\frac{.}{\vdash \{\boxed{i = k + 1}[\boxed{i + 1}/\boxed{i}]\}\ \boxed{\texttt{i}}\ \texttt{:=}\ \boxed{\texttt{i + 1}}\ \{\boxed{i = k + 1}\}}$$

We compute the substitution, and we get:

$$\frac{.}{\vdash \{i + 1 = k + 1\}\ \boxed{\texttt{i}}\ \texttt{:=}\ \boxed{\texttt{i + 1}}\ \{i = k + 1\}}$$

that is:

$$\frac{.}{\vdash \{i = k\}\ \boxed{\texttt{i}}\ \texttt{:=}\ \boxed{\texttt{i + 1}}\ \{i = k + 1\}}$$

Example 3

Is this sequent valid: $\vdash \{s = k\}\ s := s + i\ \{s = k + i\}$?

Example 3

Is this sequent valid: $\vdash \{s = k\}\ s := s + i\ \{s = k + i\}$?

We use the assignment rule:

$$\frac{\cdot}{\vdash \{\,Q\,[\,e\,/\,x\,]\}\ \texttt{x}\ \ \texttt{:=}\ \ \texttt{e}\ \{\,Q\,\}}$$

Example 3

Is this sequent valid: $\vdash \{s = k\}\ s := s + i\ \{s = k + i\}$?
We use the assignment rule:

$$\frac{\cdot}{\vdash \{Q[e/x]\}\ \mathtt{x}\ \mathtt{:=}\ \mathtt{e}\ \{Q\}}$$

to obtain:

$$\frac{\cdot}{\vdash \{s = k + i[s + i/s]\}\ \mathtt{s}\ \mathtt{:=}\ \mathtt{s\ +\ i}\ \{s = k + i\}}$$

Example 3

Is this sequent valid: $\vdash \{s = k\}\ s := s + i\ \{s = k + i\}$?
We use the assignment rule:

$$\frac{.}{\vdash \{Q[e/x]\}\ \texttt{x} := \texttt{e}\ \{Q\}}$$

to obtain:

$$\frac{.}{\vdash \{s = k + i[s + i/s]\}\ \texttt{s} := \texttt{s + i}\ \{s = k + i\}}$$

We compute the substitution, and we get:

$$\frac{.}{\vdash \{s + i = k + i\}\ \texttt{s} := \texttt{s + i}\ \{s = k + i\}}$$

Example 3

Is this sequent valid: $\vdash \{s = k\}\ s := s + i\ \{s = k + i\}$?
We use the assignment rule:

$$\frac{\cdot}{\vdash \{Q[e/x]\}\ \texttt{x := e}\ \{Q\}}$$

to obtain:

$$\frac{\cdot}{\vdash \{s = k + i[s + i/s]\}\ \texttt{s := s + i}\ \{s = k + i\}}$$

We compute the substitution, and we get:

$$\frac{\cdot}{\vdash \{s + i = k + i\}\ \texttt{s := s + i}\ \{s = k + i\}}$$

Simplifying the equation, we obtain:

$$\frac{\cdot}{\vdash \{s = k\}\ \texttt{s := s + i}\ \{s = k + i\}}$$

Example 4

Is this sequent valid: $\vdash \{i \geq 0\}\ i := i + 1\ \{i \geq 1\}$?

Example 4

Is this sequent valid: $\vdash \{i \geq 0\}\ i := i + 1\ \{i \geq 1\}$?

We use the assignment rule:

$$\frac{.}{\vdash \{Q[e/x]\}\ \mathtt{x}\ :=\ \mathtt{e}\ \{Q\}}$$

Example 4

Is this sequent valid: $\vdash \{i \geq 0\}\ i := i + 1\ \{i \geq 1\}$?

We use the assignment rule:

$$\frac{\cdot}{\vdash \{ Q [e / x] \}\ \texttt{x := e}\ \{ Q \}}$$

to obtain:

$$\frac{\cdot}{\vdash \{ i \geq 1 [i + 1 / i] \}\ \texttt{i := i + 1}\ \{ i \geq 1 \}}$$

Example 4

Is this sequent valid: $\vdash \{i \geq 0\}\ i := i + 1\ \{i \geq 1\}$?

We use the assignment rule:

$$\frac{\cdot}{\vdash \{Q[e/x]\}\ \texttt{x := e}\ \{Q\}}$$

to obtain:

$$\frac{\cdot}{\vdash \{i \geq 1[i+1/i]\}\ \texttt{i := i + 1}\ \{i \geq 1\}}$$

We compute the substitution, and we get:

$$\frac{\cdot}{\vdash \{i + 1 \geq 1\}\ \texttt{i := i + 1}\ \{i \geq 1\}}$$

Example 4

Is this sequent valid: $\vdash \{i \geq 0\}\ i := i + 1\ \{i \geq 1\}$?
We use the assignment rule:

$$\frac{\cdot}{\vdash \{Q[e/x]\}\ \texttt{x := e}\ \{Q\}}$$

to obtain:

$$\frac{\cdot}{\vdash \{i \geq 1[i+1/i]\}\ \texttt{i := i + 1}\ \{i \geq 1\}}$$

We compute the substitution, and we get:

$$\frac{\cdot}{\vdash \{i + 1 \geq 1\}\ \texttt{i := i + 1}\ \{i \geq 1\}}$$

which simplifies to:

$$\frac{\cdot}{\vdash \{i \geq 0\}\ \texttt{i := i + 1}\ \{i \geq 1\}}$$

Precondition Strengthening Rule

$$\frac{\vdash \{P'\}\ S\ \{Q\} \qquad P \rightarrow P'}{\vdash \{P\}\ S\ \{Q\}}$$

Precondition Strengthening Rule

$$\frac{\vdash \{P'\}\ S\ \{Q\} \qquad P \rightarrow P'}{\vdash \{P\}\ S\ \{Q\}}$$

- If $\vdash \{P'\}\ S\ \{Q\}$ and P implies P', then $\vdash \{P\}\ S\ \{Q\}$ holds as well
- This rule is used to strengthen the precondition by making it *less strict* while maintaining the validity of the Hoare triple

Example: Precondition Strengthening

Is this sequent valid: $\vdash \{n > 0\}\ n := n + 1\ \{n \geq 1\}$?

The precondition strengthening rule is:

$$\frac{\vdash \{\boxed{P'}\}\ S\ \{Q\} \qquad \boxed{P} \rightarrow \boxed{P'}}{\vdash \{\boxed{P}\}\ S\ \{Q\}}$$

Here is the proof (which uses the assignment rule — see Example 4):

$$\frac{\dfrac{\cdot}{\vdash \{\boxed{n \geq 0}\}\ n := n + 1\ \{n \geq 1\}} \qquad \boxed{n > 0} \rightarrow \boxed{n \geq 0}}{\vdash \{\boxed{n > 0}\}\ n := n + 1\ \{n \geq 1\}}$$

Proof Rule for Postcondition Weakening

- A dual rule for postconditions called *postcondition weakening*:

$$\frac{\vdash \{P\}\ S\ \{Q'\} \qquad Q' \rightarrow Q}{\vdash \{P\}\ S\ \{Q\}}$$

- If we can prove postcondition Q', we can always relax it to something weaker Q

Example: Postcondition Weakening

Is this sequent valid: $\vdash \{n > 0\}\ n := n + 1\ \{n > 0\}$?

The postcondition weakening rule is:

$$\frac{\vdash \{P\}\ S\ \{\ Q'\ \} \qquad Q' \to Q}{\vdash \{P\}\ S\ \{\ Q\ \}}$$

Here is the proof (which uses again the assignment rule):

$$\frac{\dfrac{\cdot}{\vdash \{n > 0\}\ n := n + 1\ \{\ n \geq 1\ \}} \qquad n \geq 1 \to n > 0}{\vdash \{n > 0\}\ n := n + 1\ \{\ n > 0\ \}}$$

Proof Rule for Composition

$$\frac{\vdash \{P\}\ S_1\ \{Q\} \qquad \vdash \{Q\}\ S_2\ \{R\}}{\vdash \{P\}\ S_1\ ;\ S_2\ \{R\}}$$

- Remark: the postcondition of $\vdash \{P\}\ S_1\ \{Q\}$ is the same as the precondition of $\vdash \{Q\}\ S_2\ \{R\}$, which could be an issue in practice because they don't always match
- This could be a little restrictive, but precondition strengthening or postcondition weakening could be used to overcome this issue

Example: Composition Rule

Is this sequent valid: $\vdash \{\texttt{true}\}\ x := 2\,;\, y := x\ \{y = 2 \wedge x = 2\}$?

We use the composition rule:

$$\frac{\vdash \{P\}\ S_1\ \{Q\} \qquad \vdash \{Q\}\ S_2\ \{R\}}{\vdash \{P\}\ S_1\,;\,S_2\ \{R\}}$$

Here is the proof (which uses the assignment rule for both statements, individually):

$$\frac{\dfrac{\cdot}{\vdash \{\texttt{true}\}\ x := 2\ \{x = 2\}} \qquad \dfrac{\cdot}{\vdash \{x = 2\}\ y := x\ \{y = 2 \wedge x = 2\}}}{\vdash \{\texttt{true}\}\ x := 2\,;\, y := x\ \{y = 2 \wedge x = 2\}}$$

Proof Rule for If Statements

$$\frac{\vdash \{P \wedge C\}\ S_1\ \{Q\} \qquad \vdash \{P \wedge \neg C\}\ S_2\ \{Q\}}{\vdash \{P\}\ \texttt{if}\ C\ \texttt{then}\ S_1\ \texttt{else}\ S_2\ \{Q\}}$$

- Suppose we know P holds before the if statement and want to show Q holds afterwards
- In the then branch, we want to show $\{P \wedge C\}\ S_1\ \{Q\}$
- In the else branch, we want to show $\{P \wedge \neg C\}\ S_2\ \{Q\}$

Example: Proof for If Statement

Is this sequent valid:

$$\vdash \{\texttt{true}\}\ \texttt{if}\ \ x < 0\ \ \texttt{then}\ \ m := -x\ \ \texttt{else}\ \ m := x\ \{m \geq 0\}?$$

We use the rule for if-statements:

$$\frac{\vdash \{\ P\ \wedge\ C\ \}\ S_1\ \{\ Q\ \} \qquad \vdash \{\ P\ \wedge \neg\ C\ \}\ S_2\ \{\ Q\ \}}{\vdash \{\ P\ \}\ \texttt{if}\ \ C\ \ \texttt{then}\ \ S_1\ \ \texttt{else}\ \ S_2\ \{\ Q\ \}}$$

Example: Proof for If Statement

Is this sequent valid:

$$\vdash \{\texttt{true}\}\ \texttt{if}\ \ x < 0\ \ \texttt{then}\ \ m := -x\ \ \texttt{else}\ \ m := x\ \{m \geq 0\}?$$

We use the rule for if-statements:

$$\frac{\vdash \{\ \boxed{P} \wedge \boxed{C}\ \}\ S_1\ \{\ \boxed{Q}\ \} \qquad \vdash \{\ \boxed{P} \wedge \neg \boxed{C}\ \}\ S_2\ \{\ \boxed{Q}\ \}}{\vdash \{\ \boxed{P}\ \}\ \texttt{if}\ \ \boxed{C}\ \ \texttt{then}\ \ S_1\ \ \texttt{else}\ \ S_2\ \{\ \boxed{Q}\ \}}$$

We show two branches:

$$\frac{\dfrac{\ldots}{\vdash \{\ \boxed{\texttt{true}} \wedge \boxed{x < 0}\ \}\ m := -x\ \{\ \boxed{m \geq 0}\ \}} \qquad \dfrac{\ldots}{\vdash \{\ \boxed{\texttt{true}} \wedge \neg \boxed{(x < 0)}\ \}\ m := x\ \{\ \boxed{m \geq 0}\ \}}}{\vdash \{\ \boxed{\texttt{true}}\ \}\ \texttt{if}\ \ \boxed{x < 0}\ \ \texttt{then}\ \ m := -x\ \ \texttt{else}\ \ m := x\ \{\ \boxed{m \geq 0}\ \}}$$

Example: Proof for If Statement (Complete)

The proof of

$\vdash \{\texttt{true}\}\ \texttt{if}\ x < 0\ \texttt{then}\ m := -x\ \texttt{else}\ m := x\ \{m \geq 0\}$:

$$\dfrac{\dfrac{\dfrac{.}{\{\, -x \geq 0 \,\}\ m := -x\ \{\, m \geq 0 \,\}} \quad \texttt{t} \wedge (x < 0) \rightarrow -x \geq 0}{\vdash \{\, \texttt{t} \wedge x < 0 \,\}\ m := -x\ \{\, m \geq 0 \,\}} \qquad \dfrac{\dfrac{.}{\vdash \{\, x \geq 0 \,\}\ m := x\ \{\, m \geq 0 \,\}} \quad \texttt{t} \wedge \neg (x < 0) \rightarrow x \geq 0}{\vdash \{\, \texttt{t} \wedge \neg (x < 0) \,\}\ m := x\ \{\, m \geq 0 \,\}}}{\vdash \{\, \texttt{t} \,\}\ \texttt{if}\ x < 0\ \texttt{then}\ m := -x\ \texttt{else}\ m := x\ \{\, m \geq 0 \,\}}$$

Proof Rule for While and Loop Invariants

Proof Rule for While:

$$\frac{\vdash \{I \wedge C\}\ S\ \{I\}}{\vdash \{I\}\ \texttt{while}\ C\ \texttt{do}\ S\ \texttt{end}\ \{I \wedge \neg C\}}$$

- To understand the proof rule for while loops, we first need the concept of a **loop invariant**
- A loop invariant I ...
  1. ...holds before the loop starts and after the loop ends
  2. ...holds after every iteration of the loop

Example: Proof for While Statement

Is this sequent valid:

$$\vdash \{i = 0\}\ \texttt{while}\ (i < n)\ \texttt{do}\ i := i + 1\ \texttt{end}\ \{i = n\}?$$

Note: n is a natural number

$$\frac{\vdash \{\,I \wedge i < n\,\}\ i := i + 1\ \{\,I\,\}}{\vdash \{\,I\,\}\ \texttt{while}\ (i < n)\ \texttt{do}\ i := i + 1\ \texttt{end}\ \{\,I \wedge \neg (i < n)\,\}}$$

In this case, the loop invariant is $I = (i \leq n)$:

Example: Proof for While Statement

Is this sequent valid:

$$\vdash \{i = 0\}\ \texttt{while}\ \ (i < n)\ \ \texttt{do}\ \ i := i + 1\ \ \texttt{end}\ \{i = n\}?$$

Note: n is a natural number

$$\frac{\vdash \{\, I \wedge i < n \,\}\ i := i + 1\ \{\, I \,\}}{\vdash \{\, I \,\}\ \texttt{while}\ \ (i < n)\ \ \texttt{do}\ \ i := i + 1\ \ \texttt{end}\ \{\, I \wedge \neg (i < n) \,\}}$$

In this case, the loop invariant is $I = (i \leq n)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\cdot}{\vdash \{\, i + 1 \leq n \,\}\ i := i + 1\ \{\, i \leq n \,\}} \qquad i \leq n \wedge i < n \rightarrow i + 1 \leq n}{\vdash \{\, i \leq n \wedge i < n \,\}\ i := i + 1\ \{\, i \leq n \,\}}}{\vdash \{\, i \leq n \,\}\ \texttt{while}\ (i < n)\ \texttt{do}\ i := i + 1\ \texttt{end}\{\, i \leq n \wedge \neg (i < n) \,\}} \qquad i \leq n \wedge \neg\, i < n \rightarrow i = n}{\vdash \{\, i \leq n \,\}\ \texttt{while}\ (i < n)\ \texttt{do}\ i := i + 1\ \texttt{end}\{i = n\}} \qquad i = 0 \rightarrow i \leq n}{\vdash \{i = 0\}\ \texttt{while}\ (i < n)\ \texttt{do}\ i := i + 1\ \texttt{end}\{i = n\}}$$

Proof of Invariant for While Loop — Part I

We prove that $i \leq n$ is the invariant for:

$$\texttt{while}\ (i < n)\ \texttt{do}\ i := i + 1\ \texttt{end}$$

Remark: $I = (i \leq n)$ must be true before and after each iteration.

The sequent we proved:

$$\vdash \{i = 0\}\, \texttt{while}\ (i < n)\ \texttt{do}\ i := i + 1\ \texttt{end}\, \{i = n\}$$

The invariant holds before the loop: The precondition $i = 0$ implies $i \leq n$

The invariant is preserved by the loop body:
We need to prove the sequent:

$$\vdash \{i \leq n \wedge i < n\}\; i := i + 1\; \{i \leq n\}$$

Proof of Invariant for While Loop — Part II

The proof is below:

$$\dfrac{\dfrac{\cdot}{\vdash \{i+1 \leq n\}\ i := i+1\ \{i \leq n\}} \qquad (i \leq n \wedge i < n) \rightarrow i+1 \leq n}{\vdash \{i \leq n \wedge i < n\}\ i := i+1\ \{i \leq n\}}$$

Therefore, $i \leq n$ still holds after the iteration, including the last iteration.

At the end of the loop, the invariant $i \leq n$ holds, but the condition of the loop does not hold anymore; so, $\neg(i < n)$ holds, that is, $i \geq n$.

From $i \leq n$ and $i \geq n$ we have $i = n$.

Summary of Hoare Logic Proof Rules

Assignment

$$\frac{\cdot}{\vdash \{Q[e/x]\}\ \texttt{x := e}\ \{Q\}}$$

Precondition Strengthening

$$\frac{\vdash \{P'\}\ S\ \{Q\} \qquad P \rightarrow P'}{\vdash \{P\}\ S\ \{Q\}}$$

Postcondition Weakening

$$\frac{\vdash \{P\}\ S\ \{Q'\} \qquad Q' \rightarrow Q}{\vdash \{P\}\ S\ \{Q\}}$$

Composition

$$\frac{\vdash \{P\}\ S_1\ \{Q\} \qquad \vdash \{Q\}\ S_2\ \{R\}}{\vdash \{P\}\ S_1; S_2\ \{R\}}$$

If

$$\frac{\vdash \{P \wedge C\}\ S_1\ \{Q\} \qquad \vdash \{P \wedge \neg C\}\ S_2\ \{Q\}}{\vdash \{P\}\ \texttt{if}\ C\ \texttt{then}\ S_1\ \texttt{else}\ S_2\ \{Q\}}$$

While

$$\frac{\vdash \{I \wedge C\}\ S\ \{I\}}{\vdash \{I\}\ \texttt{while}\ C\ \texttt{do}\ S\ \texttt{end}\ \{I \wedge \neg C\}}$$

Soundness

- It can be shown that the proof rules for Hoare logic are sound:

$$\text{If } \vdash \{P\}\ S\ \{Q\}, \text{ then } \models \{P\}\ S\ \{Q\}$$

- This means that if $\vdash \{P\}\ S\ \{Q\}$ is provable using the proof rules, then $\models \{P\}\ S\ \{Q\}$ — the triple is semantically valid
- Completeness of proof rules means that if $\{P\}\ S\ \{Q\}$ is a semantically valid Hoare triple, then it can be proven using our proof rules, i.e.,

$$\text{If } \models \{P\}\ S\ \{Q\}, \text{ then } \vdash \{P\}\ S\ \{Q\}$$

- Unfortunately, completeness does **not** hold!

Relative Completeness

- The rules for precondition strengthening and postcondition weakening contain implications $\varphi \to \varphi'$
- There are valid implications that cannot be proven (due to Peano Arithmetic which is incomplete)
- The rules still provide an important guarantee, called *relative completeness*:

  If we have an oracle for deciding whether an implication $\varphi \Rightarrow \varphi'$ *holds, then any valid Hoare triple can be proven using our proof rules.*

The sum Program in Dafny

```
method mysum(n: nat) returns (sum: int)
    requires n >= 0
    ensures sum == n * (n + 1) / 2
{
    sum := 0;
    var i := 1;

    while i < n + 1
        invariant 1 <= i <= n + 1
        invariant sum == i * (i - 1) / 2
    {
        sum := sum + i;
        i := i + 1;
    }
}
```

This is a direct translation of our Hoare Logic proof into executable, verified code!

What Dafny Automatically Verifies

When you compile this Dafny program, it automatically:

1. **Initial State:** Checks that the invariant holds after initialization
   - After `sum := 0; i := 1`, does the invariant hold?
2. **Preservation:** Verifies that the loop body preserves the invariant
   - If invariant $\wedge$ condition holds before, does invariant hold after?
3. **Postcondition:** Confirms that invariant $\wedge$ $\neg$condition implies postcondition
   - invariant implies `sum == n*(n+1)/2`?

Summary

- Deductive Program Verification — a method to prove program correctness using formal logic
- Hoare Logic — a system to reason about program correctness using Hoare triples $\{P\}\ S\ \{Q\}$
- Partial Correctness vs Total Correctness (termination!)
- The Hoare Logic Proof System: A set of rules for proving the validity of Hoare triples
- Soundness and Relative Completeness